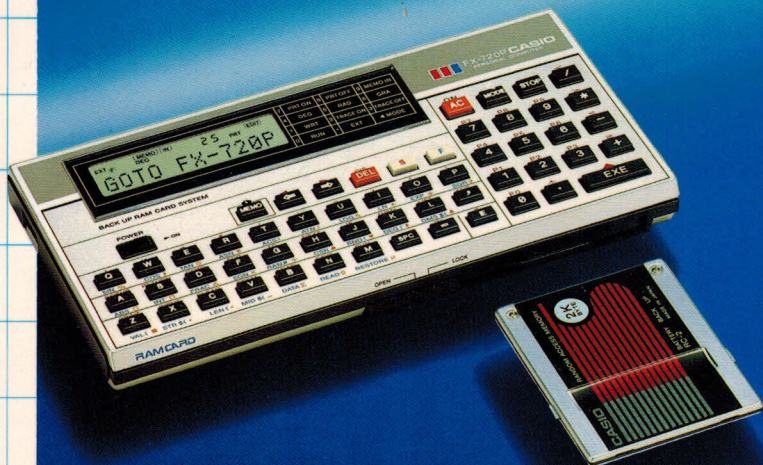
カシオポケットコンピュータ

PB-410/FX-720P

活用ハンドブック

CASIO

はじめに

本書はこれからBASICプログラムを始めようとする方にも、すでにBASICを知っていてフルに活用しようとなさる方にも、わかりやすく、すぐに使えるように説明してあります。
BASICプログラムを初めて習う方は第1章から順にお読みになり、プログラムの基本をしっかりと覚えてください。特に第3章では、プログラムの作り方やコマンドの説明をよくお読みください。なお、第3章ではプログラムを流れにそって説明しておりますので、各コマンドの書式や詳しい説明については第4章のコマンドリファレンス編を参照してください。
すでにBASICを知っている方は、第1章、第2章で基本的な操作を覚えた後は、第4章のコマンドリファレンスを読みながらお使いください。
すぐにプログラムを入力して使いたい方は、第5章の「実用ライブラリー」や、第3章の5によりPB-100のプログラムを使うことができます。
なお、本紙はPB-410とFX-720Pの両機について説明しております。両機の違いはFUNC(ファンクション)キーの部分ですので、詳しくは2ページを参照してください。
では、この本を参考にして有効に使ってください。

※表紙の写真は、はめ込み合成です。

ご使用の前に

この計算機は、カシオの高度な電子技術と品質管理のもとで、厳重な検査工程を経て、皆様のお手もとに届けられています。
本機を末ながくご愛用いただくために、次の点にご留意のうえ、お取り扱いください。

■ご使用上の注意

- 計算機は精密な電子部品で構成されていますので、絶対に分解しないでください。また、投げたり落したり等のショックや、急激な温度変化を与えないでください。特に、高温の所、湿気やホコリの多い所に放置したり保管することはしないでください。なお、温度が低いときは表示の応答速度が遅くなったり、点灯しなくなることがありますが、通常の温度になると正常にもどります。
- アダプター差し込み口には、FA-3およびFP-12S以外は接続しないでください。
- 計算機の演算中は〝-〟を表示し、この間のキー操作は一部キーを除いて無効ですから、常に表示を確認しながら、確実にキーを押してください。
- ブザーを鳴らしたときに表示が薄くなることがありますが、故障ではありません。なお、あまり表示が薄いときは、早めに電池を交換してください。
- 計算機およびRAMカードの電池は、使わない場合でも２年に１度は交換してください。
  特に消耗ずみの電池を放置しておきますと、液もれをおこし、故障等の原因になりますので、計算機内には絶対に残しておかないでください。
- 本体にはRAMは内蔵されておりませんので、必ずスロットにRAMカードを入れてご使用ください。
- 本体の電池を交換する場合には、RAMカードの内容が変化することがありますので、必ずRAMカードを本体からはずしてから電池を交換してください。
- RAMカードを取り出すスイッチを左に動かしますと、電源が切れ操作ができませんので使用中は必ずスイッチをLOCKに合わせておいでください。
- アダプター差し込み口のキャップは、本体のみで使用する場合には必ずつけて、むやみに接点には触れないでください。
- 本体およびRAMカードが強度の静電気をおびますと、メモリー内容が変化したり、キー操作ができなくなることがあります。このような場合には、一旦電池をはずし、もう一度入れなおしてください。

●オプションとの接続は本体の電源スイッチをOFFにしてから行なってください。

●計算機のお手入れは、シンナー・ベンジン等の揮発性液体をさけ、「乾いた柔らかい布」あるいは、「中性洗剤液に浸し固くしぼった布」でおふきください。

●プログラム実行中または演算中には、電源スイッチを切らないでください。

●本機は高精密機器のため、プログラム実行中に**強い振動や衝撃**を与えますと、プログラム実行が停止したり、メモリーの内容が変化する場合がありますのでご注意ください。なお、プログラムおよびデータの消失につきましては、当社ではその責任は負いかねます。

●PB-410またはFX-720Pで作成したRAMカード(プログラム)は、PB-410またはFX-720P以外では使用できません。

■保証・アフターサービス

●保証は、別紙の保証書の内容によりますので、よくお読みのうえ、記入事項を確認して、大切に保管してください。

●万一故障したときは　**❶お買い上げ店**　**❷カシオ計算機サービスセンター**のうち、ご都合のよい所へ、必ず保証書をそえて、ご持参またはご郵送ください。この場合、故障内容を具体的にお知らせください。

●修理依頼される前には、この説明書をもう一度お読みになると共に、電源の状態および、プログラムミス、操作ミスがないかをよくお調べください。

●ご不明の点やご質問、お問い合わせ等は、176ページのカシオ計算機へ直接ご連絡ください。

目　　次

目　次

目　次

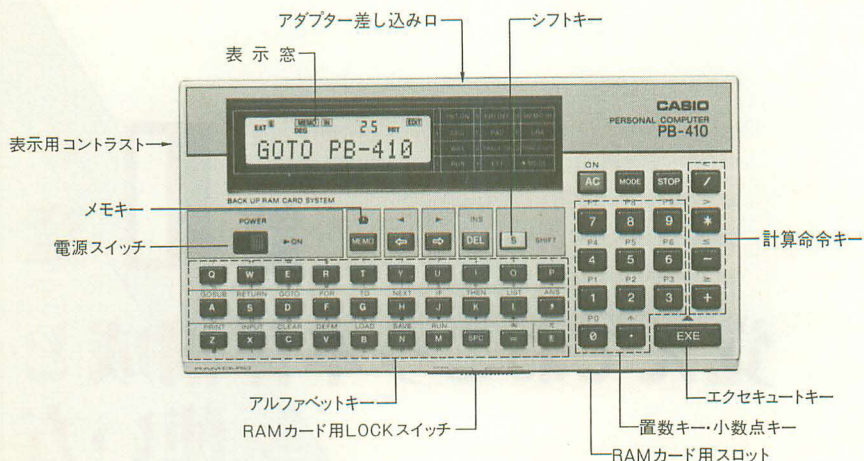
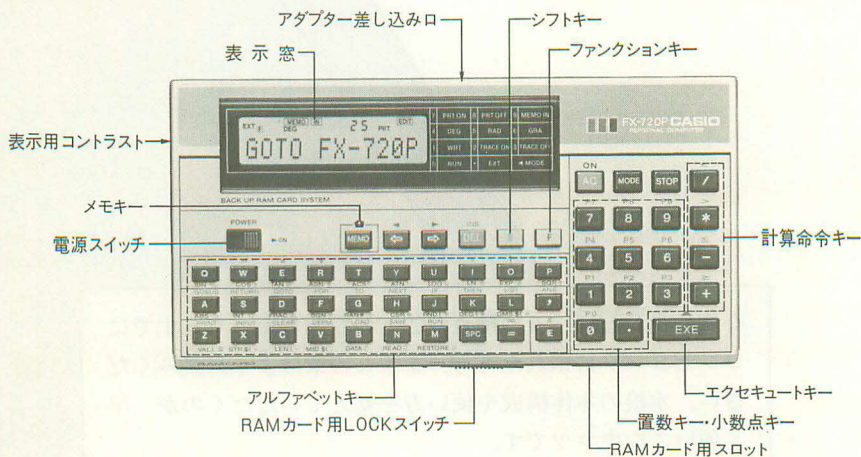
目　次

目　　次

第1章 覚えておこう本体構成と使い方

今迄にコンピュータに触れたことのない方も、もうすでにコンピュータに慣れた方も、まずこの章はよくお読みください。本機の本体構成や使い方を覚えていただくのが、早く使いこなすコツです。

1−1 各部の名称とそのはたらき

〈PB-410〉

〈FX-720P〉

PB-410とFX-720Pの違いはFX-720PにFUNC(ファンクション)キーがついている点です。このキーは、関数を入力するときにアルファベットキーと組み合わせて使います。

普通の電卓に比べて、たくさんのキーや差し込み口があります。これだけキーがたくさんあると、どのキーを何に使うのかと迷われるかもしれません。この迷いを取り除くために、これから各々のキーや差し込み口などを説明します。

●電源スイッチ

スライド式のスイッチで、右にスライドすると電源が入り、左にスライドさせると電源が切れます。

●シフトキー(赤色のⓈキー)

パネル上に赤色で書かれているワンキーコマンドや記号を表示させるときに押します。一度押しますとシフトインモードとなり"Ⓢ"が点灯します。続けて押すとシフトインモードが解除され、"Ⓢ"が消えます。

(本書ではアルファベットキーのⓈと区別するために、以後SHIFTと書きます)

●ファンクションキー(青色のⒻキー：FX-720Pのみついています)

パネル上に青色で書かれているワンタッチ関数を表示させるとき押します。一度押しますとファンクションモードとなり"Ⓕ"が点灯します。続けて押すとファンクションモードが解除され、"Ⓕ"が消えます。

(本書ではアルファベットキーのⒻと区別するために、以後FUNCと書きます)

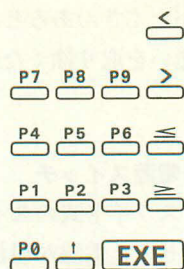
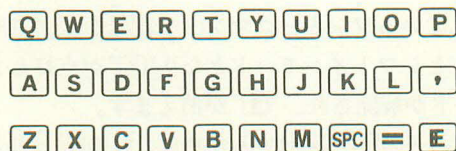
●置数キー・小数点キー、計算命令キー、エクセキュートキー

このキーの配列をよく見てください。普通の電卓と同じ配列をしていますね。この部分はちょうど四則計算(加減乗除)をするときに使いますが、少し異なる点があります。それは×(カケル)と÷(ワル)のキーがちがっていることと、=(イコール)キーがなく、EXE(エクセキュート)というキーがあります。これはコンピュータの言葉では×は＊(アスタリスク)を、÷は／(スラッシュ)を使い、=キーのかわりにEXEキーで答えを求めます。

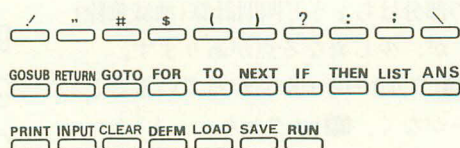
例えば、普通の電卓で12×4÷3+7−5=と操作するところを、
本機では、12＊4/3+7−5EXEと操作します。
これで、本機を普通の電卓がわりに使うこともできます。
なお、SHIFTキーに続けて押しますと、0～9キーはP0～P9のプログラムエリア指定の役目をし、・キーはべき乗計算($x^y \to x \uparrow y$)の、+−＊/キーは大小比較記号(≧、≦、>、<)を表示します。

<
P7 P8 P9 >
P4 P5 P6 ≦
P1 P2 P3 ≧
P0 ↑ EXE

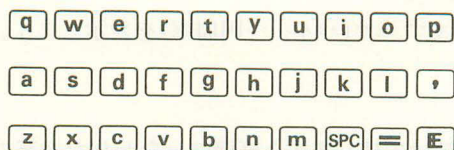
●アルファベットキー、スペースキー

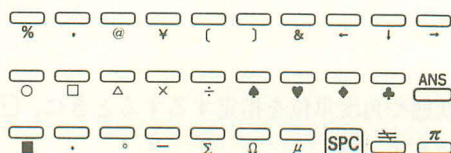
Q W E R T Y U I O P
A S D F G H J K L ,
Z X C V B N M SPC = E

このキーが本機の特長で、タイプライターのようにアルファベット26文字と、スペースキー(SPC)が並んでいます。このキーを使って命令を与えたり、プログラムを書き込んだりします。また、A～Zまでの26文字のキーはそれぞれがメモリー(記憶するところ)の役をします。
なお、A～Zのキーには別の役割があり、SHIFTキーに続けて押すと記号やBASICのコマンドを表示します。

例) SHIFT A→GOSUB、 SHIFT U→?

! " # $ ( ) ? : ; \
GOSUB RETURN GOTO FOR TO NEXT IF THEN LIST ANS
PRINT INPUT CLEAR DEFM LOAD SAVE RUN

このアルファベットキーはほかにも使い方があります。それは拡張モード(MODEキーに続いて・キーを押す。EXT点灯)での使い方で、直接押すとアルファベットの小文字を、SHIFTキーに続けて押すと特殊記号を表示します。

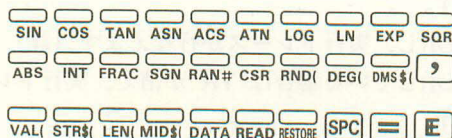
拡張モードでの働き

q	w	e	r	t	y	u	i	o	p
a	s	d	f	g	h	j	k	l	,
z	x	c	v	b	n	m	SPC	=	E

拡張モードでSHIFTキーに続けて押したときの働き

%	'	@	¥	[	]	&	←	↓	→
○	□	△	×	÷	♠	♥	♦	♣	ANS
■	·	°	─	Σ	Ω	μ	SPC	≒	π

拡張モードを解除して、アルファベット大文字に戻すには、もう一度MODE・と押します。

FX-720PにはFUNCキーがついていますので、FUNCキーに続けて押すと以下のような関数命令を表示します。

例) FUNC Q→SIN

SIN	COS	TAN	ASN	ACS	ATN	LOG	LN	EXP	SQR
ABS	INT	FRAC	SGN	RAN#	CSR	RND(	DEG(	DMS$(	,
VAL(	STR$(	LEN(	MID$(	DATA	READ	RESTORE	SPC	=	E

また、拡張モードでは英大文字を表示します。

このように、アルファベットキーはいくつもの顔を持っていますので、よく覚えておいてください。

●イコールキー(=)

このキーは計算の答を求めるためのキーではなく、代入文(40ページ参照)やIF文(62ページ参照)での判断のために使います。

なお、SHIFTキーに続いて押しますと、≠(等しくない)の記号が表示されます。

● 指数部置数キー・パイキー(π/E)

このキーは直接押すと指数部置数キーとなり、指数部(10の何乗)を置数する前に押します。例えば1.23×10^4の場合は[1][・][2][3][E][4]と押します。 指数部が負の場合は、このキーに続いて[−]キーを押します。7.41×10^{-9}は [7][・][4][1][E][−][9]と押します。

[SHIFT]キーに続いて押しますとπ(パイ……円周率)を表示します。

● アンサーキー(ANS/・)

このキーは[SHIFT]キーに続けて押しますと直前に行なわれた計算の答を覚えているキーで、マニュアル計算とプログラム計算で行なわれた計算の答を表示します。

● モードキー([MODE])

このキーは計算機の状態や角度単位を指定するするときに、[・][0]〜[9]のキーと組み合わせて使います。

[MODE][・]……"EXT"を表示し、拡張モードとなり、英小文字・特殊記号が使えます。再び押すと拡張モードを解除します。

[MODE][0]……"RUN"を表示し、マニュアル計算およびプログラム計算が行なえます。

[MODE][1]……"WRT"を表示し、プログラムの書き込みおよびチェック、編集が行なえます。

[MODE][2]……"TR"を表示し、実行トレースが行なえます。(詳しくは57ページ)

[MODE][3]……"TR"が表示している場合は"TR"が消え、実行トレース機能が解除されます。

[MODE][4]……"DEG"を表示し、角度の単位を〈度〉に指定します。

[MODE][5]……"RAD"を表示し、角度の単位を〈ラジアン〉に指定します。

[MODE][6]……"GRA"を表示し、角度の単位を〈グレード〉に指定します。

[MODE][7]……"PRT"を表示し、プリンタ接続時はプリント出力をすることができます。

[MODE][8]……"PRT"が表示している場合は"PRT"が消え、プリント出力が解除されます。

[MODE][9]……"[MEMO][IN]"を表示し、メモインモードとなります。(データバンク活用ハンドブック参照)このモードを解除するには、[MODE][0]と操作します。

●**メモキー(MEMO)**

データバンクを使うときに押します。RUNモード(MODE 0と押す)やメモインモード(MODE 9と押す)で直接押して順番に呼び戻すか、文字を指定した後に押して呼び戻すときに使います。

●**カーソルキー(◀▶)**

このキーは表示されている文字を訂正するときに便利なキーで、カーソル(表示窓で点滅している"_"のことです)を左右に移動させます。1回押すと一文字分移動し、押し続けますと文字の書いてある範囲内を続けて移動します。

●**オールクリアーキー(AC)**

このキーは全てのキーの中で一番強いキーで、どんな表示でも消してしまいます。また、エラーになって停止したときやオートパワーオフ（10ページ参照）で表示が消えているときにも押します。プログラム実行中は、プログラムを中断させます。

●**デリート・インサートキー(INS/DEL)**

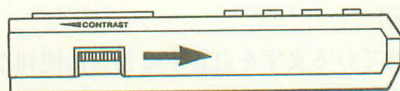
このキーも表示されている文字を訂正するときに便利なキーで、カーソルが点滅している文字を削除(デリート)します。削除した後はカーソルより右側の文字を左につめます。

また、SHIFTキーに続けて押しますと、カーソルの点滅している文字以降を右にずらし、空白をあけます。

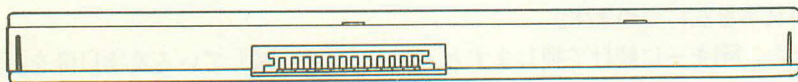
●**ストップキー(STOP)**

このキーはプログラム実行中に使うキーで、プログラムの実行を一時的に停止させます。プログラムを続けて実行させたいときはEXEキーを押すことにより再開します。

●表示用コントラスト

電池の消耗や表示窓を見る角度により、濃く見えたり薄く見えたりします。このようなときには、本体左側面にあるボリウムで、見やすい濃さに調整してください。

ボリウムは矢印方向に回すと濃くなり、逆に回すと薄くなります。
なお、最も濃い位置にしてもまだ表示が薄い場合は、電池がかなり消耗していることが考えられますので、なるべく早く電池を交換してください。

●オプション差し込み口

この差し込み口は別売のオプションを接続するコネクターで、プリンタを使うときは<FP-12S>を、テープレコーダーで記録するときには<FA-3>をつなぎます。

この差し込み口には<FP-12S>、<FA-3>以外は接続しないでください。
また、オプションを接続しないときは、必ず付属のコネクターカバーをつけて使用してください。
(オプションの使い方については92ページを参照してください。)

1−2 電源について

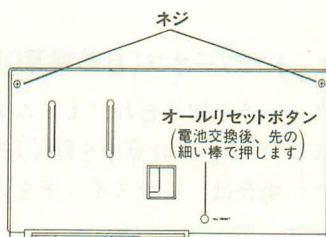
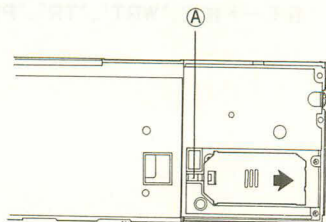
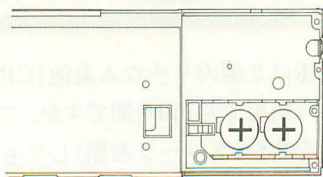
本体は2個のリチウム電池(CR-2032)を電源としています。電池の寿命は本体のみ使用で約140時間ですが、ブザーを多用すると短かくなります。コントラストを調整(8ページ参照)しても表示が薄いときは電池が消耗していますので、早めに交換してください。電池は必ず2個ともいっしょに交換してください。

※電池は2年以上使用した場合、液もれをおこす危険がありますので使わない場合でも2年に1度は必ず電池交換してください。

※本体に組み込まれている電池はモニター用の電池ですので所定の時間に満たないうちに消耗することがあります。なるべく早目に交換してください。

■電池交換の方法

本体にRAMカードがセットされている場合は、RAMカードを必ずはずしてから交換してください。電池交換が終ったらRAMカードをスロットにセットし直してください。(13ページ参照)

① 電源スイッチを切ってから、裏面の2個のネジをはずし、裏ブタをはずします。(ドライバーはなるべく先の細い精密ドライバーをお使いください。)

② 図のⒶを押しながら電池押さえ板を矢印方向にスライドさせ、はずします。

③ 古い電池を2個とも取り出します。
(電池ボックスを下に向けて軽くたたけば簡単にはずれます。)

④ 新しい電池の表面を乾いた布などでふいてから、⊕側を上にして入れます。⊕⊖をまちがえないように注意してください。

⑤ 電池押え板をはずしたときと反対方向にスライドさせてとじます。

⑥ 裏ブタをネジ止めします。

※消耗ずみの電池は絶対に火中に投下しないでください。
破裂することがあり非常に危険です。

電池は、幼児の手のとどかないところに保管してください。
万一、飲み込んだ場合にはただちに医師と相談してください。

■オートパワーオフ(自動電源OFF)

スイッチの切り忘れによるムダな電力消費を防ぐ自動節電機能で、操作完了後(プログラム計算中を除く)約6分で自動的に電源オフになります。
この場合は、電源スイッチを入れ直すか、ACキーを押せば、再び電源オンになります。

※電源オフになってもメモリー内容およびプログラム内容は消えませんが、角度指定や各モード指定("WRT"、"TR"、"PRT"等)はすべて解除されます。

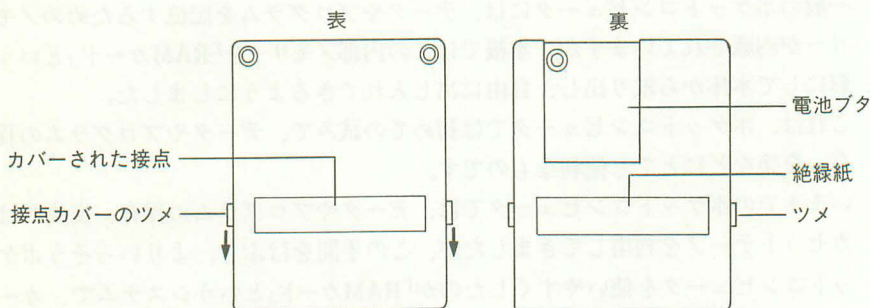
1－3 RAMカードについて

■RAMカードの特長

一般のポケットコンピュータには、データやプログラムを記憶するためのメモリーが内蔵されていますが、本機ではこの内部メモリーを「RAMカード」という形にして本体から取り出し、自由に出し入れできるようにしました。
これは、ポケットコンピュータでは初めての試みで、データやプログラムの保存、交換などにとても便利なものです。
いままでのポケットコンピュータでは、データやプログラムの保存、交換にはカセットテープを利用してきましたが、この手間をはぶき、よりいっそうポケットコンピュータを使いやすくしたのが「RAMカード」というシステムで、カートリッジ単位でデータやプログラムを簡単に交換して処理できますから大変便利になりました。RAMカードに記憶された内容は内蔵電池でバックアップされていますから、本体からはずしてもその内容が消えてしまうことはありません。
RAMカードにはRC-4(4Kバイト)、RC-2(2Kバイト)の2種類があります。
データやプログラムをそれぞれのRAMカードに入力しておけば、必要な時に必要な場所でそのカードを装着して処理することができますから、活用範囲がグンと広がるわけです。

※本機は、本体内にRAMエリアを持っていませんので、RAMカードが装着されていない状態では使用できません。

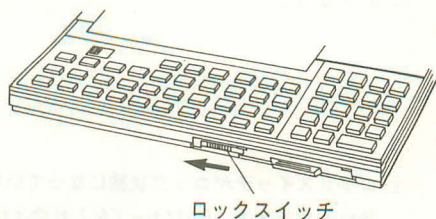
■取り扱いの注意点

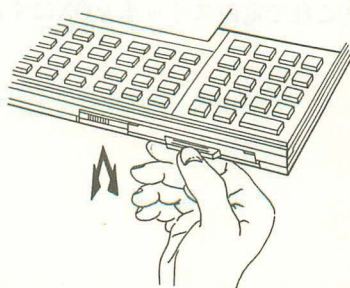
本機専用のRAMカードには、RC-4(4Kバイト)とRC-2(2Kバイト)の２種類がありますが、取り扱い方法は同じです。

- 接点に触れないでください。
  左右のツメを持って矢印方向に引くと接点が露出しますが、この接点に指または金属物などで触れると、RAMカードが使用できなくなることがあります。本体からはずしたら必ずカバーをして接点を保護してください。
- RAMカードに強度の静電気を加えると、記憶している内容が変化したり、本体のキー入力ができなくなる場合があります。このようなときはRAMカードの電池を一度はずして入れ直してください。(このとき、記憶されている内容は消去されます。)
- RAMカードは分解したり、ヒネリ、曲げなどの力を加えることは絶対にしないでください。
- RAMカードを本体からはずして保管されるときは、付属のケースに入れて、ゴミ・ホコリや直射日光の当らない所に保管してください。
- RAMカードは、メモリーバックアップ用のリチウム電池を内蔵しています。電池を取り出すと、それまでに記憶されていた内容は消えてしまいます。
  電池交換を行なう前には、カードの記憶内容を一度カセットテープなどに記憶させ、電池交換が終ったら再び記憶内容をRAMカードに記憶させます。
  (92ページ参照)
- 絶縁紙は、万一逆さに入れたときに接点を守るためにはってありますので、絶対にはがさないでください。
- RAMカードは必ず電池を入れてご使用ください。
- RAMカードはカシオのRAMカード専用機以外には使用しないでください。

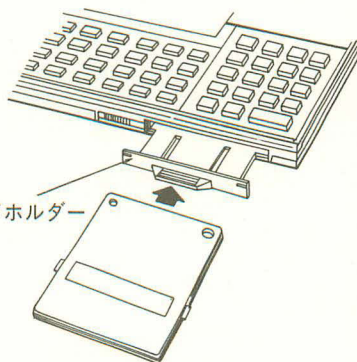
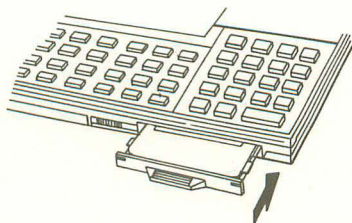
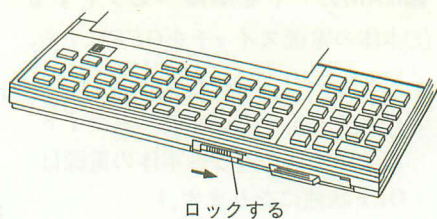
■RAMカードを本体へセットする

①本体の電源スイッチをOFFにする。

②ロックスイッチを左側にスライドさせます。(このとき本体の電源はOFF状態になります。)

注)ロックスイッチをスライドせずに、無理にカードホルダーを引き出すと、ロックスイッチが破損しますので注意してください。

③カードホルダーの凸部を下にかるく押しながら、かるく引き出します。

注)カードホルダーは途中までしか引き出せません。無理に引き出そうとすると破損しますので注意してください。

④RAMカードを表面(接点のある面)を上にして接点カバーをしたままカードホルダーに差しこむます。

⑤RAMカードが水平になるようにカードホルダーの凸部を下へかるく押し、完全に納めます。

⑥カードホルダーを少し上に押し上げるようにして矢印方向にカチッと音がして完全に停止するまで押し込みます。

⑦RAMカードのロックスイッチを右側にスライドさせロックすると、電源スイッチをONにする待機状態になります。

注）**ロックスイッチがロック状態になっていないと、本体の電源スイッチをONにしても電源が入りません。**RAMカードを入れ換えたりしたときは忘れずにロックしてください。

⑧これで電源スイッチをONにすればいつでも使えます。

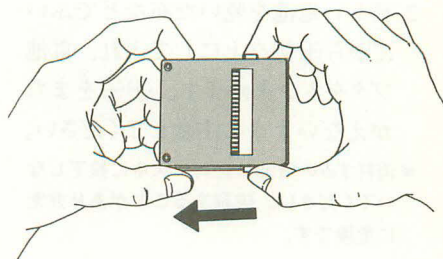
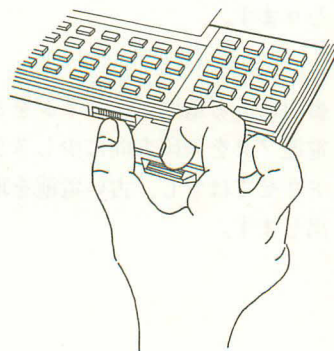
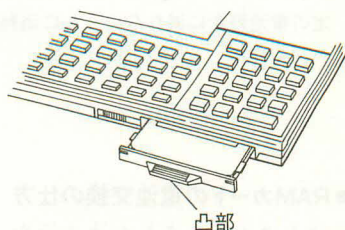
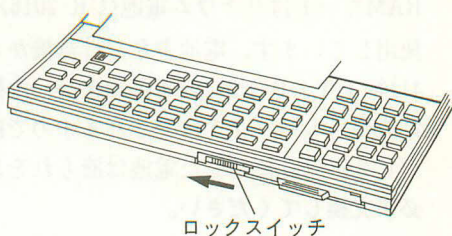
■RAMカードのはずし方

①本体の電源スイッチをOFFにします。

②ロックスイッチ左側にスライドさせ、ロックを解除します。

③カードホルダーの凸部を下へ押しながら、かるく引き出します。

④カードホルダーの凸部を下へかるく押えながら、RAMカードの両端を持って引き出します。このとき、接点に触れないように気をつけてください。

⑤取り出したRAMカードは接点が露出した状態になっていますので、すぐに接点カバーの両側のツメをスライドさせて接点をカバーしてください。

注)使用しない場合は、必ずRAMカード付属ケースに入れて保管してください。

別のカードを交換してセットする場合は、前項のRAMカードのセット方法(④~⑦)を参考に行なってください。

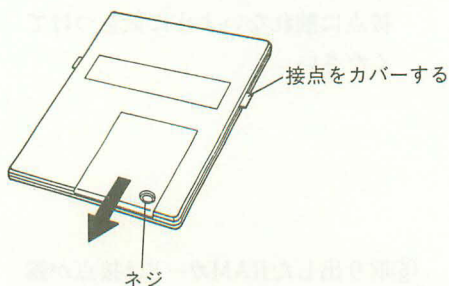
■RAMカードの電池(メモリーバックアップ用)交換

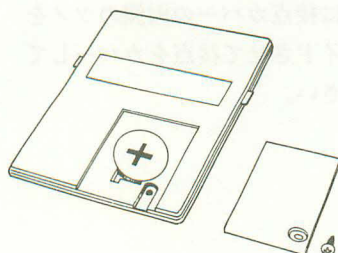
RAMカードはリチウム電池<CR-2016>1個をメモリーバックアップ電源として使用しています。電池寿命は計算機からはずして使わずに保管した場合、RC-4は約1年、RC-2は約2年ですが、本機にセットして使用した場合は、本体の電源からもバックアップされますのでRC-4の電池寿命は約2年に延びます。ただし2年以上使用した電池は液もれをおこす場合がありますので、**2年以内に必ず交換してください。**

※付属のRAMカード(RC-2)には、工場出荷時にすでに電池が組み込まれてありますので、所定の電池寿命に満たないうちに消耗することがあります。なるべく早目に電池を交換してください。

●RAMカードの電池交換の仕方

接点にカバーをしたまま行なってください。接点に触れますと故障の原因となります。

1. 裏面にある電池ブタのネジをとり、電池ブタを矢印方向に少しスライドさせてはずし、古い電池を取り出します。

2. 新しい電池を乾いた布などでふいてから⊕側を上にして入れ、電池ブタをネジ止めます。⊕⊖をまちがえないように注意してください。

※消耗ずみの電池は絶対に火中に投下しないでください。破裂することがあり非常に危険です。

電池は、幼児の手のとどかないところに保管してください。
万一、飲み込んだ場合にはただちに医師と相談してください。

RAMカード内のプログラムおよびデータは、電池にて保護されていますので、必ず電池寿命以内に電池交換を行なってください。
なお、電池寿命が切れたときや、電池交換時には、プログラムおよびデータは消えますので、大切なプログラムおよびデータはカセットテープに記録しておいてください。

■ユーザーエリアとシステムエリア

RAMカード内のRAMエリアはRC-2で2048バイト、RC-4で4096バイトとなっいます。このエリアは大別して3つに分けられています。
1つはプログラムや変数を管理する**システムエリア**、もう1つはAからZまでの変数に使われる**固定変数エリア**、そしてこの2つを除いたプログラムやデータバンクに使われる**フリーエリア**です。

RAMカード	システムエリア	固定変数エリア	フリーエリア
RC-2	272バイト	208バイト	1568バイト
RC-4	272バイト	208バイト	3616バイト

実際にプログラムを組んだり、データバンクに記憶させていくのはフリーエリアで、この範囲内を自由に使うことができます。

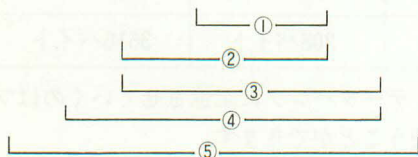
1−4 計算の前に

■計算の優先順位

計算には「優先順位」という規則があり、たし算・ひき算よりかけ算・わり算の方を先に計算することになっています。本機はこの優先順位を計算機自身が自動的に判別するようにできています。この機能はとても便利で、数式をそのまま打ち込んでいけば正しい答が求められます。
計算の優先順位は次のように定められています。

①関数（SIN, COS等）
②べき乗（↑）
③×(＊)、÷(／)
④＋、−

計算はこの優先順位に従って行なわれますが、優先順位が同じときは頭から(左から)、カッコが使用されたときはカッコの内が最優先されます。

〈例〉 2＋3＊SIN (17＋13) ↑2＝2.75

■入出力桁数と演算桁数

本機の数値を入力できる範囲(入力桁数)は仮数部12桁、指数部２桁です。内部演算も同じで、仮数部12桁、指数部２桁で行なわれます。
数値を表示できる範囲(出力桁数)は通常、仮数部10桁で、マニュアル計算による答の表示とプログラム中での答の表示は異なります。マニュアル計算では指数部や負符号がつく場合、仮数部、指数部、負符号を含めて12桁まで表示します。プログラム中では仮数部10桁、指数部２桁を全て表示しますが、合計が12桁をこえる場合、最初に頭から12桁分を表示し、以後順に表示が左に送られるようにして表示されます。

〈例〉

マニュアル計算

入力	表示
1 [・] 2345678912 [EXE]	1.234567891
12345678912 [*] 100 [EXE]	1.2345678 E12
12345678912 [*] −100 [EXE]	−1.234567 E12

プログラム計算

PRINT　12345678912 [*] −100 の場合

表示の外に消えます。	表示	まだ表示されていません。
	−1.234567891	E12
−	1.234567891E	12
−1	.234567891 E1	2
−1.	234567891E12	

自動的に送られます。

第2章 まずは動かそう

ここでは、本機に慣れ親んでいただくために、とにかくさわってください。多少まちがった操作をしても壊れるものではありません。習うより慣れろのことわざ通りに、まずは簡単な操作から。

2−1 とにかくさわってみよう

ここではまずさわって動かしてみて、どのように動くかを見てみましょう。
まず最初にすることは、本機を手に取って、電源スイッチを右にスライドさせて、電源を入れます。

すると表示は次のようになるはずです。

RUN DEG
READY P0

初めにこの表示を消してみます。ACキーを押してください。〝READY P0〟は消えましたね。このとき左端に点滅している〝_〟があります。これを**「カーソル」**と呼び、ここから文字を書き始めます。

このカーソルが点滅している状態を**「入力待ち状態」**とも言い、計算や命令を待っているのです。カーソルは通常〝_〟の点滅ですが、文字を続けて書いていくうちに、〝▮〟の点滅となることがあります。本機では1行に書ける文字数が62文字までですので、56文字以上書きますと注意信号として表われます。なお、表示窓の上に〝RUN〟と〝DEG〟が点灯していると思いますが、これは状態表示といい、今どのような状態になっているかを示します。〝RUN〟はRUN(ラン)モードを示し、マニュアル計算やプログラムの実行が行なえます。〝DEG〟は角度単位で、度になっていることを示します。角度単位はほかに、MODE 5と押して指定するラジアンモード(〝RAD〟点灯)、MODE 6と押して指定するグレードモード(〝GRA〟点灯)があります。この角度単位は三角関数などを使うときに必要となります。電源スイッチONでは〝DEG〟が表示されます。
状態表示はほかにもMODE 1と押すプログラム書き込みモード(〝WRT〟点灯)、MODE 2と押すトレースモード(〝TR〟点灯、57ページ参照)、MODE 7と押すプリントモード(〝PRT〟点灯、96ページ参照)、MODE 9と押す電子メモ入力モード(〝MEMO IN〟点灯、MODE ・と押す拡張モード(〝EXT〟点灯)を示すものがあります。

このような状態表示はさわっているうちに覚えてきますので、ここでは見るだけにしてもかまいません。

では、計算機にさわって表示をさせてみましょう。
もしいろいろさわって、状態表示が色々と点灯している時は、一度電源スイッチを切ってから再び電源を入れてください。

まず初めに簡単な計算をしてみましょう。

例） 123+456=579

ACを押します。
数式の通りにキーを押します。

1 2 3 + 4 5 6

この後に =のかわりにEXEで答を求めます。

EXE

`_`

`123+456_`

`579`

どうでしたか、計算が簡単にできますね。

では、次にもう少し長い計算をしてみましょう。

例） 45×6+89=359

ここでは45をまちがえて46と押してしまったとします。

4 6 * 6 + 8 9

`46*6+89_`

45を46と押してしまったことに気がつきました。でも、あわてずに、カーソルキー(←)でカーソルをまちがった所に合わせます。

← ← ← ← ← ←

`46*6+89`

カーソルと6とが交互に点滅します

ここで正しい5のキーを押します。

5

`45*6+89`

これで計算式が正しくなりましたので、答を求めます。

EXE

`359`

このように、途中でまちがいに気付いたときは、カーソルキーを使って簡単に訂正することができます。ただし、EXEキーを押してしまった後では、最初から入れなおしてください。

それでは、アルファベットキーを使って文字を書いてみましょう。
アルファベットキーはタイプライターと同じ配列になっており、この配列をASCII型配列といいます。現在のポケコンはこのASCII型配列のキーボードが主流ですので、最初は慣れないでしょうが、だいたいの位置は覚えておいてください。

まず、文字を書いてみましょう。

例）文字は「ABCXYZ」とします。

ABCと書いてみます。

A B C　　　　ABC_

次にXYZと書いてみます。

X Y Z　　　　ABCXYZ_

それでは、ABCとXYZの間を1文字分あけてみます。
カーソルをXに合わせます。

← ← ←　　　　ABCXYZ

1文字分あけます。

SHIFT INS/DEL　　　　ABC_XYZ

文字間をあけるときは、あけたい箇所の次にカーソルを合わせ、SHIFT INS/DELで1文字分あきますので、何文字分もあけたいときは、繰り返し同じ操作をします。

本機には数字・アルファベットのほかに、いくつかの特殊文字と呼ばれる文字があります。これはゲームや科学記号に便利な文字です。特殊文字の種類については5ページを参照してください。

ここで少し表示させてみましょう。

例）♤♡◇♧のマークを表示させる。

まず拡張モードを指定します。

AC MODE ・　　　　点灯します EXT _

このマークはSHIFTキーに続けてアルファベットキーを押します。

SHIFT ♠/H SHIFT ♥/J SHIFT ♦/K SHIFT ♣/L　　　　♠♥♦♣_

例）ΣΩμの記号を表示させる。

拡張モードになっているので、そのままSHIFTキーに続けて押します。

SHIFT Σ/B SHIFT Ω/N SHIFT μ/M　　　　♠♥♦♣ΣΩμ_

このようなマークや記号が用意されていますので、色々と利用して使ってみてください。なお、拡張モードになっているときに、今迄のアルファベット大文字を表示するモードに戻す場合は、もう一度MODE・と押し、"EXT"を消します。
これでキーの押し方はだいたいわかったかと思います。このように色々とさわっているうちに"ERR 2"が表示されて、キーを押しても動かなくなることがあります。これは故障ではなく、まちがった命令をしましたというメッセージで、**「エラーメッセージ」**と呼ばれるものです。このときはあわてずに、ACキーを押せば表示が消えて、また動くようになります。このようなエラーメッセージにはいくつかありますので、詳しくは52ページまたは169ページをご覧ください。

2－2 データバンクはとても便利

本機の特長としてデータバンク機能があります。このデータバンク機能はMEMOキーを使うだけで簡単にメモデータを記憶させたり呼び戻したりすることができ、少し応用を加えるだけで色々な使い方があります。

たとえば　電話帳
　　　　　時刻表
　　　　　スケジュール表
　　　　　各種早見表　　など……

また、BASICプログラムの中から検索、呼び出し、書き込みもできますので、さらに利用範囲を広げることができます。

たとえば　得意先リスト
　　　　　製品リスト
　　　　　見積計算
　　　　　図書文献メモ
　　　　　ゴルフ集計　　など……

もっと色々な使い方もあると思います。
このデータバンクの使い方や応用例につきましては、別冊の「データバンク活用ハンドブック」をご覧ください。

2−3 まず始めに基本計算

ここでは簡単な四則計算ぐらいを行なってみますが、関数電卓を使ったことのない方は注意していただきたいのです。本機は完全数式通りというたし算、ひき算よりかけ算、わり算を先に計算する機能を持っているからです。

例1) 23+4.5−53=−25.5
操作) 23 + 4.5 − 53 EXE

-25.5

※ここからは、数字キーはワクをはずして記します。

例2) 56×(−12)÷(−2.5)=268.8
操作) 56 * − 12 / − 2.5 EXE

268.8

※負符号がつく場合は、数字の前に − キーを押します。

例3) 7×8−4×5=36
操作) 7 * 8 − 4 * 5 EXE

36

※かけ算を先に計算してから、ひき算を計算します。

例4) $(4.5\times10^{75})\times(-2.3\times10^{-78})=-0.01035$
操作) 4.5 E 75 * − 2.3 E − 78 EXE

-0.01035

※指数部は E キーに続いて押します。

もう一つの計算として、メモリーを使った代数計算があります。この計算は、ある一定の数値を色々と計算するときに便利です。

例えば　$3x+5=$
　　　　$4x+6=$
　　　　$5x+7=$

というような計算があり、xの値が123.456であるとき、同じ数値を繰り返し押すのは面倒なものです。この計算を手間をかけずに行なう方法はないでしょうか。解決策は変数と呼ばれるメモリーを使うことです。この例では代数計算にxという代数を使っていますので、変数Xを使って計算します。

まず、変数Xに123.456を入れます。

[X][=] 123.456 [EXE]

この[=]は等しいという意味ではなく、変数Xに123.456を入れるという意味です。
では、計算をしてみましょう。

キー操作	表示
3 [*] [X] [+] 5 [EXE]	375.368
4 [*] [X] [+] 6 [EXE]	499.824
5 [*] [X] [+] 7 [EXE]	624.28

こんなに簡単にできます。

本機にはこのような変数がA～Zまで26個ありますので、色々な数値を覚えておくことができます。
この例では変数Xの数値は一定で、計算式が異なりますが、逆に計算式が一定で、変数の値が異なる場合はどうでしょう。
もし、計算式が"3 x + 5 ="と決っていて、xの値が123、456、789と変化する場合、今のような方法では操作が面倒になります。実際には計算式を計算機が覚えてくれて、変数Xの値だけを変えればよいのです。この便利な計算方法を**「プログラム計算」**といいます。本機はこのプログラム計算が得意なのです。ここではプログラムを使う前のマニュアル計算を行なっていますので、プログラムについては、第3章のプログラム編をご覧ください。

2—4 関数計算もおてのもの

本機は一般の四則計算のほかに、関数計算の機能も持っています。
この関数機能はプログラム中に組み込んでも使えますが、ここではマニュアルでの使い方について説明します。

本機に組み込まれている関数は次の通りです。

関数名		書　式	引数範囲
三角関数	sin x	SIN x	度：$\|x\|<1440$
	cos x	COS x	ラジアン：$\|x\|<8\pi$
	tan x	TAN x	グレード：$\|x\|<1600$
逆三角関数	$\sin^{-1} x$	ASN x	$\|x\|\leqq 1$
	$\cos^{-1} x$	ACS x	$\|x\|\leqq 1$
	$\tan^{-1} x$	ATN x	
平方根	$\sqrt{x}$	SQR x	$x\geqq 0$
常用対数	log x	LOG x	$x>0$
自然対数	ln x	LN x	$x>0$
指数関数	e^x	EXP x	$-10^{10}<x\leqq 230.2585092$
べき乗	x^y	$x \uparrow y$	$x<0$のとき、yは自然数
10進→60進		DMS $($x$)※	$\|x\|<10^{100}$、但し変換は$\|x\|<100000$
60進→10進		DEG(x,y,z)※	$\|$DEG(x,y,z)$\|<10^{100}$
整数化		INT x	
整数部除去		FRAC x	
絶対値化	$\|x\|$	ABS x	
符号化	(正数→1 0→0 負数→-1)	SGN x	
四捨五入	(xの10^yを四捨五入)	RND(x,y)※	$\|y\|<100$
乱数		RAN#	

※DMS $、DEG、RNDは引数を必ず(　)でくくります。

では、関数を使って計算をしてみましょう。

● **三角関数(sin、cos、tan)、逆三角関数(sin⁻¹、cos⁻¹、tan⁻¹)**

三角・逆三角関数を使うときは、必ず角度単位の指定(DEG、RAD、GRA)を行なってください。(角度単位を変更しない場合は、新たに行なう必要はありません。)

〈例〉 sin 12.3456°=0.2138079201

〈操作〉 [MODE][4]→"DEG"

SIN 12.3456[EXE]　　　`0.2138079201`

※ここからは、アルファベットキー、数字キーはワクをはずして記します。

※FX-720Pでは[FUNC][SIN]と押しても同じです。

〈例〉 cos 63°52′41″=0.4402830847

〈操作〉 COS DEG[SHIFT][(]63[,]52[,]41[SHIFT][)][EXE]

`0.4402830847`

〈例〉 2・sin45°×cos65.1°=0.5954345575

〈操作〉 2[×]SIN 45[×]COS65.1[EXE]　　　`0.5954345575`

〈例〉 $\sin^{-1}0.5=30°$

〈操作〉 ASN 0.5[EXE]　　　`30`

〈例〉 $\cos(\frac{\pi}{3}\text{rad})=0.5$

〈操作〉 [MODE][5]→"RAD"

COS[SHIFT][(][SHIFT][π][/]3[SHIFT][)][EXE]　　　`0.5`

〈例〉 $\cos^{-1}\frac{\sqrt{2}}{2}=0.7853981634\text{rad}$

〈操作〉 ACS[SHIFT][(]SQR2[/]2[SHIFT][)][EXE]　　　`0.7853981634`

〈例〉 tan(−35gra)=−0.612800788

〈操作〉 [MODE][6]→"GRA"

TAN[−]35[EXE]　　　`-0.612800788`

● **対数関数(log、ln)、指数関数(e^x、x^y)**

〈例〉 $\log 1.23(=\log_{10}1.23)=0.0899051114$

〈操作〉 LOG 1.23[EXE]　　　`0.0899051114`

〈例〉 ln 90(=$\log_e$ 90)=4.49980967

〈操作〉 LN90 EXE

4.49980967

〈例〉 e^5=148.4131591

〈操作〉 EXP5 EXE

148.4131591

〈例〉 $10^{1.23}$=16.98243652
(常用対数1.23の真数を求める)

〈操作〉 10 SHIFT ^ 1.23 EXE

16.98243652

〈例〉 $5.6^{2.3}$=52.58143837

〈操作〉 5.6 SHIFT ^ 2.3 EXE

52.58143837

〈例〉 $123^{\frac{1}{7}}$(=$\sqrt[7]{123}$)=1.988647795

〈操作〉 123 SHIFT (SHIFT (1 / 7 SHIFT) EXE

1.988647795

〈例〉 log sin40°+log cos35°=−0.278567983
その真数は……0.5265407845(sin40°×cos35°の対数計算)

〈操作〉 MODE 4 → "DEG"
LOG SIN 40 + LOG COS 35 EXE
10 SHIFT ^ SHIFT ANS EXE

−0.278567983
0.5265407845

● その他の関数(√、SGN、RAN#、RND、ABS、INT、FRAC)

〈例〉 $\sqrt{2}+\sqrt{5}$=3.65028154

〈操作〉 SQR 2 + SQR 5 EXE

3.65028154

〈例〉 正数であれば"1"を、負数であれば"−1"を、0であれば"0"を与える。

〈操作〉 SGN 6 EXE
SGN 0 EXE
SGN − 2 EXE

1
0
−1

〈例〉 乱数発生(0<RAN#<1の擬似乱数)

〈操作〉 RAN SHIFT # EXE

0.790373907б

〈例〉　12.3×4.56の答を10^{-2}の位で四捨五入する。
12.3×4.56=56.088

〈操作〉　RND SHIFT ⌐ 12.3 ✱ 4.56 , − 2 SHIFT ⌐ EXE

※ RND(x, y)のときのyは$|y|<100$

`56.1`

〈例〉　|−78.9÷5.6|=14.08928571

〈操作〉　ABS SHIFT ⌐ − 78.9 ∕ 5.6 SHIFT ⌐ EXE

`14.08928571`

〈例〉　$\frac{7800}{96}$の整数部は……81

〈操作〉　INT SHIFT ⌐ 7800 ∕ 96 SHIFT ⌐ EXE

`81`

※この関数は元の数値をこえない最大の整数を求めます。

〈例〉　$\frac{7800}{96}$の小数部は……0.25

〈操作〉　FRAC SHIFT ⌐ 7800 ∕ 96 SHIFT ⌐ EXE

`0.25`

● **有効桁数指定、小数以下指定**

有効桁数と小数以下の指定は"SET"コマンドにより行ないます。

有効桁数指定……SET　E n　(n=0~8)
小数以下指定……SET　F n　(n=0~9)
指定解除…………SET　N

※マニュアル計算での有効桁数指定の場合の"SET E 0"は8桁指定となります。

※指定を行なうと、指定桁の下1桁目を四捨五入して表示します。
なお、計算機内部やメモリー内にはもとの数値が残っています。

〈例〉　100÷6=16.66666666………

〈操作〉　SET　E4 EXE（有効桁数4桁指定）
100 ∕ 6 EXE

`1.667E01`

〈例〉　123÷7=17.57142857……

〈操作〉　SET F2 EXE（小数以下2桁指定）
123 ∕ 7 EXE

`17.57`

〈例〉　1÷3=0.3333333333………

〈操作〉　SET N EXE（指定解除）
1 ∕ 3 EXE

`0.3333333333`

●10進↔60進変換(DEG、DMS $)

〈例〉　14°25′36″＝14.42666667

〈操作〉 DEG [SHIFT](14 [,] 25 [,] 36 [SHIFT]) [EXE]

```
14.42666667
```

〈例〉　12.3456°＝12°20′44.16″

〈操作〉 DMS [SHIFT]$ [SHIFT](12.3456 [SHIFT]) [EXE]

```
12° 20′ 44.16
```

〈例〉　sin　63°52′41″＝0.897859012

〈操作〉 [MODE] [4]

SIN DEG [SHIFT](63 [,] 52 [,] 41 [SHIFT]) [EXE]

```
0.897859012
```

関数計算もこのように簡単にできます。

第3章
BASICプログラミング

本章では、主にBASICのプログラムとはどのようなことか、どのように作るのかについて説明してあります。
今迄、プログラムに慣れていない方は、この章を繰り返しお読みになって、基本をマスターしてください。

3−1 プログラムとは？

「**プログラム**」と聞くとよく難しいものだと感じる方がいますが、プログラムにも簡単なものから難しいものまであり、代数式を記憶させ、その代数式に数値を代入して計算させるのも立派なプログラムです。
まずは難しい考え方は抜きにして、楽な気持ちで話を進めていきましょう。

3−1−1 プログラムはこんなに便利

私達の身のまわりには、色々な計算があります。仕事で使う会計や金融計算、測量や計測、その他にも家計簿や経費の計算などと考えられるだけでも多いものです。この計算も1回だけで終るものならかまいませんが、何度も繰り返し同じ計算式で数値を変えて計算するのも大変な手間です。こんな計算は、本機にとって最も得意な仕事です。たとえば、$y=2x^2+5x+13$という数式があるとします。この中でxの値が変化するときのyの値を求めるとすれば、xの値をかえながら同じ計算をしなければなりません。この手間をはぶくために、この式を記憶させてみましょう。
まずは見てください。

```
10  INPUT X
20  Y=2*X↑2+5*X+13
30  PRINT Y
```

これが数式を記憶させたプログラムです。
細かい説明は後にするとして、2行目は計算式をそのまま書いています。これでも立派なプログラムで、計算が便利になります。
このようにプログラムとは、けっして難かしいものばかりではなく、身近な計算式をそのまま記憶させるだけでもかなり便利に使えます。
それでは、プログラムについて順に説明していきましょう。

3−1−2 プログラムの仕組み

まず、プログラムの仕組みを覚えてください。

```
10  INPUT X
20  Y=2*X↑2+5*X+13
30  PRINT Y
```

これは前に出てきたプログラムです。このプログラムを分解して仕組みを見てみましょう。

このプログラムは大きく分けると3つに分けられます。

```
10 INPUT X ························入力部
20 Y=2*X↑2+5*X+13 ··········計算部
30 PRINT Y ························出力部
```

最初の入力部はデータ(計算に必要な数値など)を計算機に入れる(入力する)部分です。2つ目の計算部は計算をさせて答を求める部分で、一番の要となります。最後の出力部は計算機に答えてもらう(表示させる)部分で、計算をしただけでは何も答をおしえてくれませんので、答を表示させます。
計算機はなんでもしてくれますが、正しい命令を与えてやらなければ何もしてくれません。そのために、**データを入れなさい**(入力部)という命令と、**答を表示しなさい**(出力部)という命令を記憶させるのです。
この3つの部分をさらに細かく分解すると、次のようになります。

```
10    INPUT    X
行番号  コマンド  オペランド
```

行番号とは、プログラムの流れにそってつける順番を示すもので、この行番号の小さい方から順に計算機が読んで、実行していきますから、実行させたい順につけてください。なお、この例のように、10、20、30……と10刻みに書いているのは、後で追加が必要となったときに便利にするためです。本当は1、2、3……と続けてもよいのですが、1.5とか、12.3のように小数点以下はつけられません。
行番号の次に続くのが**コマンド**で、計算機にどのようなことをさせるかの命令を与えます。このコマンドは色々あり、命令により使い分けます。本当は全部のコマンドを覚えていただきたいのですが、一度に覚えるのは大変ですので、最低限必要なものを覚えていただいて、少しづつ数を増していってください。
コマンドの種類や機能については、103ページからの「コマンドリファレンス」を参照してください。
コマンドの後の**オペランド**は、コマンドの持つ命令を補足するためにあり、つくものとつかないものがあります。この例ではコマンドが"INPUT"ですので、データを入力しなさいという意味があり、その入力したデータを入れるメモリーを指定するのがオペランドで、この場合変数Xに入れなさいという意味になります。
次の行では、

```
20    Y=2*X↑2+5*X+13
行番号      代入文
```

行番号は前に続けて20としています。次の代入文は＝(イコール)の右辺の値を左辺の変数に入れる(代入する)という意味です。この代入文も実際はコマンドとして"LET"がつき、

```
20 LET Y=2*X↑2+5*X+13
```

とすれば"LET"がコマンドで、代入文がオペランドということになります。行番号30は、行番号10と同じで、行番号、コマンド、オペランドでできています。

```
30    PRINT    Y
|       |      |
行番号  コマンド  オペランド
```

プログラムとは、だいたいこのような仕組みでできており、これ等のコマンドやオペランドが多くなったり、行数が増えたりして大きなプログラムとなります。

3−1−3 プログラムは簡単だ

プログラムの仕組みさえ知ってしまえば、プログラムなんてたいしたものではないことがおわかりだと思います。
プログラムを作るには、入力部、計算部、出力部の三本の柱がわかれば充分に活用できます。全部のコマンドを一度に覚える必要はないのです。まずは簡単な"INPUT"、"PRINT"、それに計算の代入文を使って、身近な計算をやらせてみるのです。
プログラムを早く覚えるコツは、ただ漠然とプログラムを覚えようとせずに、身近な問題を探してプログラムにしてみるのです。会社で繰り返し行なう伝票計算や金種計算、測量や設計の計算、家での家計簿やオーディオの時間計算など、考えてみれば色々とあるはずです。この問題の中で計算式にしやすいものを探して、プログラム化してみるのです。ただ漠然と覚えようとせずに、1つの目的を持ってそのプログラムを作り、それに必要なコマンドをまず覚えるのです。あとは、この作業を繰り返しながらプログラムをより便利なものへと改良していけばよいのです。
もう一つのコツは、全体を一度に作ろうとはせずに、部分的に作り、あとで一つにするのです。どんなに大きくて複雑なプログラムも、部分的に分ければよりわかりやすくなるはずです。
これ等のことを考えながら、少しずつでいいですから、ゆっくりと行なってください。基本さえマスターすれば、後はコマンドリファレンス編で各コマンドの機能を読みながら使うだけで立派にプログラムが作れます。
それでは、プログラムを作ってみましょう。

3−2 プログラムの作成

ここからがBASICプログラミングの本題で、実際にプログラムを作り上げます。

3−2−1 フローチャート(流れ図)を作る

フローチャートという名前はあまりなじみのない名前かもしれません。これは作業の順番を図式化したもので、流れ図ともいいます。

よく「BASICにはフローチャートはいらない」という言葉を聞きますが、それは頭の中でフローチャートが描ける人が言う言葉で、慣れないうちは全体の作業の段取りがわからないものです。そのために、簡単なフローチャートを描いて流れをつかむことが大切です。

フローチャートを描くにも正式には専門の記号がありますが、そんな記号は覚えずに、1つの作業ごとに □ で囲んで線で継ぐだけでよいのです。

では実際にプログラムを作りながら説明していきましょう。

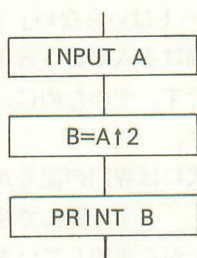
例として、ある数を入力して、その数の2乗を求めるプログラムを作ってみましょう。まず、部分的に考えますと、計算部があります。そして、ある数を入力するのですから入力部があります。それに、答を表示させるわけですから出力部があります。この3つを □ で囲むと次のようになります。

この3つの要素を順につなげるわけで、最後にくるのは答の表示だということがすぐにわかると思います。次に、答を求めるために計算をしなければなりません。計算をするためには、データを入力してやらねばなりません。

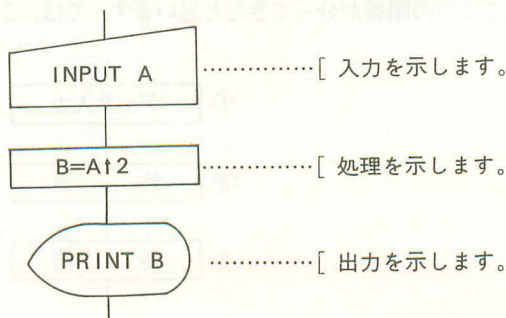
これで3つの順番が分ってきたと思います。では、この3つをつなげてみましょう。

これでフローチャートはでき上りましたが、このフローチャートをさらにプログラムに近い形にしてみましょう。

1番目はデータを入力する命令です。これはINPUT文を使い、変数にデータを入力させますので、変数を決めます。ここでは、仮に変数Aを使いますので、①の内容は、〝INPUT A〟となります。２番目の計算は、入力された変数Aの内容を２乗し、答を別の変数に入れます。もう一つの変数をBとしますと、〝B=A↑2〟となります。この計算式は**代入文**と言い正式には〝LET B=A↑2〟と書きますが、LETは省略できますので〝B=A↑2〟だけでもかまいません。そして、3番目では答を表示させます。PRINT文を使い、答の入っている変数Bの内容を表示させますので、〝PRINT B〟となります。この3つの作業をもう1度フローチャートにします。

```
INPUT A
   |
B=A↑2
   |
PRINT B
```

この後は、３つの作業に行番号をつければプログラムのでき上りです。

```
10 INPUT A
20 B=A↑2
30 PRINT B
```

このように、フローチャートを作りながら順に組み立てていった方が、プログラムを早く、簡単に作ることができます。
通常は、この２つのフローチャートの内の１つだけ作ればよいのですが、慣れないうちは1つ目のフローチャートを作るようにした方が便利です。
実際のフローチャートには正式な記号があり、それぞれ意味があります。この記号についてはあまり意識しなくてもかまいませんが、この例を正式な記号で描いてみましょう。

これ等の記号は巻末に記載してありますので、参照してください。

3−2−2 プログラムを作る

これまでは、プログラムを作る前の段取りを話してきましたが、ここからは実際にプログラムを作ってみましょう。

まず簡単な例として、２つの数値を入れて、その２つの数値の和・差・積・商を求めてみましょう。

ここでも重要なものは３本の柱である入力部、計算部、出力部で、この考え方に当てはめフローチャートを作ってみましょう。

最初に２つの数値を入力するわけですから、INPUT文を使います。INPUT文はキーボードからデータを変数に入力する命令です。ここで１つ考えることは、データや答を入れておく変数です。プログラムを作っていく上で迷いがちなのが変数の使い方です。この変数はＡからＺまでのアルファベット１文字の名前を持つものと、Ａ(３)などのように添字とよばれる要素を伴う配列変数があります。これ等の変数の中から選び出すわけですが、慣れないうちは、出てくる順番にＡ、Ｂ、Ｃ、……と選べばよいと思います。

ここでは２つの数値を入れますので、ＡとＢを選び、〝INPUT A,B〞とします。このINPUT文はカンマ(,)で区切ることにより、１つでいくつもの変数を扱えます。

それでは、次の計算部はどうでしょうか。ここでは４つの計算を行なうわけですから、答も４つ出てきます。この４つの答を入れる変数をここではＣ、Ｄ、Ｅ、Ｆとします。

まず加算は、Ａ＋Ｂですので、C=A+Bとします。

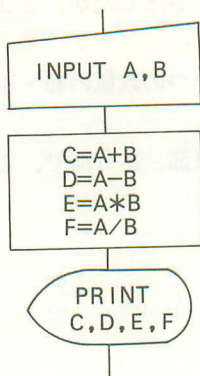
　減算は、Ａ－Ｂですので、D=A−Bとします。

　積算は、Ａ＊Ｂですので、E=A*Bとします。

　除算は、Ａ／Ｂですので、F=A/Bとします。

これで、計算部はできました。次は、この答を表示させます。

表示させる命令はPRINTですから、〝PRINT C,D,E,F〞としましょう。

このプログラム化した型式を再びフローチャートにしてみますと、次のようになります。

これに行番号をつけてプログラムを完成させます。

```
10 INPUT A,B
20 C=A+B
30 D=A−B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
```

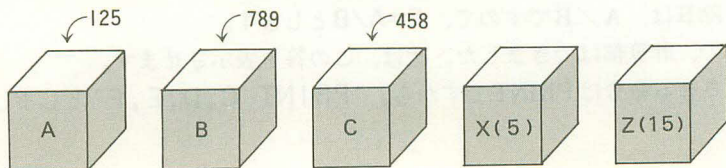
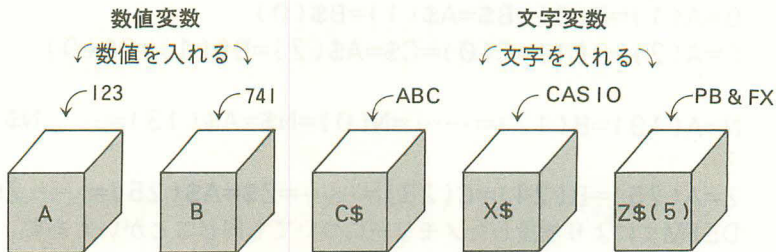
そして、最後にプログラムの終了を示す"END"をつけます。

```
70 END
```

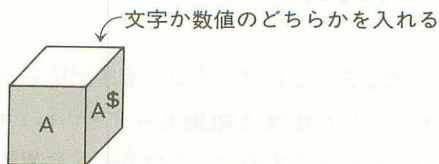
これがプログラムが完成しました。このように順をおって組み立てていくと、プログラムも簡単に作れます。最初から色々な命令を使ってむずかしく考えるよりも、簡単でも使えるプログラムを作りましょう。

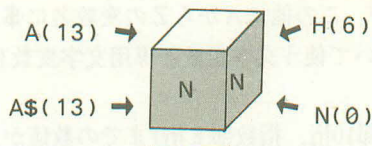
ワンポイントレッスン

変数とは

プログラムを作る上で重要な要素として、変数があります。変数とは、入力したデータや計算の答を入れておく箱で、各々に名前がついています。この変数には標準のAからZまでの変数と、配列変数と呼ばれAからZの名前に区別をする添字のついたA(5),B(50)という変数があります。

この変数にはさらに2種類あり、数値を入れる数値変数と文字列を入れる文字変数があります。今迄でできた変数は計算をするための数値を入れておくので、数値変数でしたが、この他にAからZの変数名に$(ドルマーク)をつけたA$,B$,C$と書いて使う文字変数と専用文字変数($)という特別な、文字変数があります。
数値変数には10桁(仮数部10桁、指数部2桁)までの数値が入り、文字変数には7文字までの文字列が入ります。また、専用文字変数には30文字までの文字列が入ります。

この2種類の変数は入れるものが異なりますので、数値変数に"ABC"のような文字を入れることはできませんし、文字変数に数値を入れることもできません。
これ等の変数は用途により使い分け、計算のための数値を入れるのであれば数値変数を使い、メッセージや記号を入れるのであれば文字変数を使います。
配列変数は、多くの変数を使いデータを記憶していくときに便利で、1番目、2番目……というように順番で示される番号で変数を区別します。配列変数については、プログラム中で使いながら説明していきますので、ここでは変数の種類として覚えておいてください。

〈変数使用上の注意〉

本機では、数値変数と文字変数は変数名が同じであれば同じ箱を使っているのです。

このため、1つのプログラム内でAという数値変数とA$という文字変数を同時に使えません。

また、配列変数(76ページ参照)を使うときも同じ箱を色々な名前で呼ぶことがありますので、重ねて使わないように注意してください。

すべて同じ箱の名前です。

```
A=A(0)=A$=A$(0)
B=A(1)=B(0)=B$=A$(1)=B$(0)
C=A(2)=B(1)=C(0)=C$=A$(2)=B$(1)=C$(0)
⋮
N=A(13)=B(12)=……=N(0)=N$=A$(13)=……N$(0)
⋮
Z=A(25)=B(24)=C(23)=……=Z$=A$(25)=……Z$(0)
```

DEFM文により増設したメモリーについても同じことがいえます。

3-2-3 プログラムの入力

ここでは、組み立てたプログラムを実際に憶えさせて(入力して)みましょう。入力するプログラムは、前に出てきた2つの数値を入力して四則計算をさせるプログラムを使います。

プログラム

```
10 INPUT A,B
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
70 END
```

このプログラムを記憶させるわけですが、まず記憶させる準備をします。

電源スイッチをオンにした状態は**RUNモード**といい、マニュアル計算やプログラムの実行を行なうモードですので、このモードを**WRTモード**(プログラム記憶モード)に切り換えます。プログラムを記憶させることを別のいい方で「書き込む」ともいいますので、WRTモードは**「書き込みモード」**ともいいます。

では、WRTモードに切り換えるために、MODE 1と押してください。

プログラムを記憶させていない状態では上のような表示になります。
上の4桁の数字は残りステップ数で、RC-2装着時は最大1568ステップ、RC-4装着時は最大3616ステップを示しています。この数字はプログラムを記憶させたり、メモリーを増設したり(78ページ参照)することにより減ってきます。下の0から9はプログラムエリアの使用状態で、点滅している数字が現在指定されているプログラムエリアです。このままプログラムを記憶させると、P0のプログラムエリアに記憶されます。プログラムエリアはP0からP9までの10個のエリアに分かれています。
もし、何かのプログラムが記憶されている場合は、下の数字が表示せずに、"_"というカーソルが表示されます。ここでは全部のプログラムを消して、P0のプログラムエリアに記憶させますので、

NEW ALL EXE

と操作してください。この操作は全プログラムを消す命令で、P0にプログラムを記憶させる準備にもなります。
それでは、キーを操作してプログラムを記憶させます。
記憶は次の手順で行ないます。

1 0 SHIFT INPUT A , B EXE ——一行ごとの最後に必ず押してください。
（SHIFT INPUT）——アルファベットキーを1つづつ押して I N P U T としても同じです。

2 0 C = A + B EXE
3 0 D = A − B EXE
4 0 E = A * B EXE
5 0 F = A / B EXE
6 0 SHIFT PRINT C , D , E , F EXE
7 0 E N D EXE

以上のキー操作でこのプログラムが記憶されました。
なお、EXEキーを押した後に行番号や次のコマンド、代入文との間に1文字分空白が開きますが、これは表示上見やすくしてくれるだけで、プログラム実行上は関係ありません。
プログラムの記憶が正しくできましたか？
もし、うまくできなくてもあわてずに、ゆっくりと確実にキーを押してください。
まちがえても、次のように操作すれば簡単に訂正できます。

● EXEキーを押すまえにまちがいに気付いた

このときは、EXEキーを押さずに、⇦⇨キーを使ってまちがえた箇所にカーソルを合わせ、訂正をします。

例1) `10 INPUT S_` "A"と押すところと"S"と押してしまった。

⇦キーを1回押して"S"にカーソルを合わせる。

⇦ `10 INPUT S`

正しいキーを押します。

A `10 INPUT A_`

続けて正しい操作をしてEXEキーを押す。

, B EXE `10 INPUT A,B`

例2) `RINT C,,E,F_` 行番号60の"D"を抜かしてしまった。

⇦キーを4回押して、挿入したい箇所の次に合わせます。

⇦⇦⇦⇦ `60 PRINT C,,`

1文字分開けます。

SHIFT INS `60 PRINT C,_`

"D"を入れます。

D `0 PRINT C,D,`

訂正が終ればEXEキーを押します。

EXE `60 PRINT C,D`

例3) `40 EE=A*B_` "E"を1文字多く入れてしまった。

⇦キーを5回押して"E"に合わせます。

⇦⇦⇦⇦⇦ `40 EE=A*B`

1文字分削除しますので、DELキーを1回押します。

DEL `40 E=A*B`

削除が終りましたら、EXEキーを押します。

`40 E=A*B`

●EXEキーを押してからまちがいに気付いた。

EXEキーを押した後ではプログラムとして記憶されてしまいますので、"LIST" コマンドを使って再び呼び出し、正しく訂正します。

例）　行番号50を"50　F=A/N"とまちがえてしまった。

LISTコマンドで行番号50を呼び出す。

SHIFT LIST　50 EXE　　　50 F=A/N_

L I S T と押しても同じです

⇦キーを1回押して、"N"に合わせます。

⇦　　　50 F=A/N

正しいキーを押します。

B　　　50 F=A/B_

訂正が終りましたら、必ずEXEキーを押します。

EXE

もし行番号60以降が記憶されていれば行番号60が表示されます。　　　60 PRINT C,D

このほかに訂正がなければACキーを押して表示をクリアーします。

AC

EXEキーを押しても、このようにLISTコマンドで呼び出して訂正ができます。なお、新たに行番号をつけて書き直すこともできます。すでにある行番号をつけて記憶させた場合は、後から記憶させた方が優先され、前に記憶させた行は消えます。

このようにしてプログラムを記憶させますが、記憶の操作が終了したら、MODE 0と押してRUNモードにします。WRTモードのままでは記憶させたプログラムを消したり、書きかえてしまったりすることがありますので、記憶の操作終了後は必ずRUNモードにしてください。

ワンポイントレッスン

プログラムエリア

本機の特長としてプログラム分割機能があります。これはプログラムエリアを10個にわけ、独立したプログラムを記憶させておくことができます。
プログラムエリアはP0、P1、P2、……P9とあり、使い方は同じですが各々独立しています。たとえば3本のプログラムを良く使う場合に、分割機能がなければ、そのつど別のプログラムをテープから読み込んだり、RAMカードを取りかえなければなりません。しかし、プログラムエリアをわけて使えれば、3本のプログラムをP0、P1、P2と記憶させておくことができます。
このプログラム分割機能はとても便利なものですが、1つ注意していただきたいのがステップ数です。プログラムエリアをわけて記憶させても、全エリアの使用ステップ数の合計が総ステップ数(RC-2使用時1568ステップ、RC-4使用時3616ステップ)以内でなければなりません。

プログラムエリアの指定はSHIFTキーに続いて0から9のキーを押すことにより行なえます。プログラム指定はRUNモードとWRTモードの両方でも行なえますが、RUNモードで指定した場合はそのプログラムエリアに記憶されているプログラムが自動的にスタートします。WRTモードで指定した場合はプログラムはスタートせずに、プログラムエリアの指定のみ切り換わります。
プログラムエリアの区別は、はっきりとしてください。プログラムを記憶させるときやカセットテープに記録したり、カセットテープから呼び戻したりするときに、別のプログラムエリアで操作しますと正しく行なえません。
電源スイッチをオンにしたときや、ACキーによるオートパワーオフ解除ではP0のエリアが指定されています。
確認の仕方はMODE0と押して、"READY"の後に出ている数字により確認できます。

例)　MODE0→READY　P3……プログラムエリア　P3

3−2−4 プログラムの実行

では、前に記憶させたプログラムを使って、実際に計算させてみましょう。
[MODE][0]と押してRUNモード("RUN"点灯)であることを確認してください。
プログラムを実行させるには2つの方法があります。

①プログラムエリア指定による実行
[SHIFT]キーに続いて[0]〜[9]の数字キーを押しますと、プログラムエリアの指定となり、プログラムが記憶されていれば、プログラムをスタートさせます。
例) [SHIFT][P0]

②RUNコマンドによる実行
　プログラム実行開始コマンド"RUN"により実行開始
例) [SHIFT][RUN] ([R][U][N]としても同じ)[EXE]
この2つの実行方法の違いは、①の方法では必ずプログラムエリアの先頭から実行を開始し、②の方法では先頭からも、任意の行番号からの実行もできることです。
まず、①の方法で実行します。

操　作	表　示
[SHIFT][P0]	?
ここで2つのデータを入力します。	
例) 45 [EXE]	?
36 [EXE]	81　STOP

PRINT文を実行すると、"STOP"が点灯します。

2つのデータを入力すると、和が表示されます。次の答を表示させるには[EXE]キーを押します。

操　作	表　示
[EXE]	9　STOP
[EXE]	1620　STOP
[EXE]	1.25　STOP

今度はRUNコマンドにより実行してみましょう。
ただし、RUN EXE と操作すると①の方法と同じ結果になりますので、20行目から実行してみましょう。

操　作

SHIFT RUN 20 EXE　　　81

（RUN 20 EXE でも同じです）

EXE　　　9

EXE　　　1620

EXE　　　1.25

このように、RUNコマンドでの実行は、行番号を指定しなければ先頭から実行を開始し、行番号を指定すれば、指定行番号から実行を開始します。
実際にはこの2つの実行方法にはもう一つの違いがあります。
次の操作をしてみてください。

操　作

SHIFT P5　　　_

P5のエリアにプログラムが記憶されていなければ、何も表示されません。
ここでRUNコマンドにより実行させてみます。

SHIFT RUN EXE　　　_

（RUN EXE でも同じです）

RUNコマンドの実行では、現在設定されているプログラムエリア（この例ではP5)のプログラムを実行させます。もしこのP5に設定されているときに、P0のプログラムを実行したい場合はどうすればよいでしょうか。
それは、SHIFT P0 と操作して、プログラムエリアを指定するのです。

SHIFT P0　　　?

このように、2つの実行方法には違いがありますので、用途に合わせて使い分けてください。
プログラムを作り上げ、記憶させた後、すぐにでも実行させてみたいものです。もし、実行させてエラー(ERR表示)になっても、がっかりしないでください。こんなときには、次の項目を読んでまちがい探し(デバッグと言う)を行なってください。

ワンポイントレッスン

ステップ数のかぞえ方

本機の持っているプログラム容量は、全部で<RC-2>セット時は1568ステップ、<RC-4>セット時は3616ステップです。
このステップとはプログラムを記憶できる許容量を示す単位で、プログラムを記憶させていくと、残りステップ数が減っていきます。
現在の残りステップ数は、MODE 1と押してWRTモードにすると表示されます。

例）

MODE 1

WRT 1568 ——残りステップ数

P 0123456789

また、使用するステップ数は次のようにかぞえます。

- 行番号…………1～9999のいくつであっても2ステップ
- コマンド………1ステップ
- 関数……………1ステップ
- 文字……………1文字で1ステップ
- 以上の他に、区切りとしてEXEキーを押して記憶させるのに1ステップ必要とします。

- メモリーを増設した場合は、1つにつき8ステップ必要とします。

例）初期状態……26メモリー、1568ステップ
　　DEFM 10 EXE……36メモリー、1488ステップ

3−2−5 デバッグ(まちがいを直す)

プログラムを作り上げ、いざ実行開始というときに、実行させてもエラーが表示されたり、結果が思うように得られない。このような事はよくあることですので、がっかりせずに、原因を追求してください。

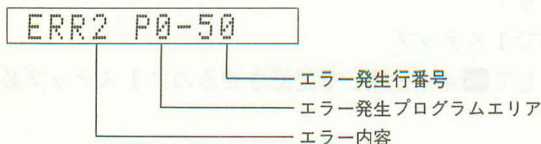
プログラム中にある「**まちがい**」のことを「**バグ**」(虫の意味) と呼び、この虫を取り除くという意味で「**デバッグ**」といいます。

このデバッグの方法も原因により異なります。実行中にエラー表示になる場合と、エラーにはならないが結果が思うように得られない場合です。

実行中にエラーが表示される場合は、エラーの発生した箇所とエラーの内容を知らせてくれますので、原因追求は比較的簡単にできますが、エラーは表示しないが結果が思うように得られない場合は、けっこう厄介なものです。

では、順を追ってデバッグをしてみましょう。

(1) エラー表示によるデバッグ

エラー表示とは、**エラーメッセージ**とも呼ばれ、

このように、3つの要素をおしえてくれます。

エラー内容は、"ERR"に続くコードナンバーにより、どのようなエラーが起きたかを知らせてくれます。このコードナンバーは1から9の数字で、"ERR1"は「メモリーオーバー」とか、"ERR2"は「構文エラー」というような内容が決っています。このコードナンバーの内容については、巻末169ページの「エラーメッセージ一覧表」を参照してください。

エラー発生プログラムエリアは、エラーの発生したプログラムエリアを知らせてくれます。

エラー発生行番号は、エラーの発生した行番号を知らせてくれます。

この3つの要素により「**どこで**」、「**どのような**」エラーが起きたのかを知ることができます。

それでは例を上げ説明していきましょう。

よく起きるエラーに"ERR2"があります。これは「**構文エラー**」といい、プログラムを記憶させるときに、まちがって記憶させてしまうと起きるのです。

前の例題で使ったプログラムをまちがえて記憶させてみます。

操　作	表　示
MODE 1	P _123456789
SHIFT LIST 20 EXE	20 C=A+B_
⇦ ⇦ DEL	20 C=AB
EXE	30 C=A-B_
AC MODE 0	READY P0

ここでは、行番号20の〝C＝A＋B″を〝C＝AB″とまちがえました。
では実行させましょう。

操　作	表　示
SHIFT P0	?
45 EXE	?
12 EXE	ERR2 P0-20

ここでエラーメッセージが表示されます。これは、プログラムエリアP0の行番号20で、構文エラーが発生しましたという意味です。
では、行番号20を調べてみましょう。

操　作	表　示
AC……エラー解除	_
MODE 1	P _123456789
SHIFT LIST 20 EXE	20 C=AB_

ここで正しいプログラムと合っているかをチェックします。
AとBの間の〝＋″が抜けていますので、正しく直します。

操　作	表　示
⇦	20 C=AB
SHIFT INS	20 C=A_B
＋ EXE	30 C=A-B
AC MODE 0	READY P0

このように、〝ERR2″はプログラムの入力ミスが多いので、〝ERR2″が表示されたときはエラーの発生した行番号のプログラムをよくチェックしてください。
なお、記憶のまちがいだけでなく、READ文(81ページ参照)でデータを読み込むときに、数値変数に文字データを読み込もうとすると〝ERR2″が表示されます。〝ERR2″の発生した場所にREAD文があるときは、DATA文中のデータもチェックしてください。
では、エラーの種類別にチェックポイントを上げてみましょう。

●ERR 1：メモリー不足。スタックオーバー。
残りステップ数を確認して、DEFM文でステップ数以上をメモリーに変換しようとしたかチェックする。

●ERR 2：構文エラー
記憶させたプログラムにまちがいがないかチェックする。

●ERR 3：数学的エラー
数式の演算結果が10^{100}以上であったり、関数の入力範囲をこえていないかをチェックする。特に変数を使っている場合は、前後関係から変数の内容をチェックする。(0による除算や、負数の平方根をとっている場合が多い。)

●ERR 4：未定義行番号エラー
GOTO文やGOSUB文、RESTORE文による行番号の指定が正しくないので、行番号を確認する。

●ERR 5：引数エラー
引数やパラメータを必要とするコマンドや関数において、引数やパラメータの値をチェックする。特に変数を使っている場合は、前後関係から変数の内容をチェックする。

●ERR 6：変数エラー
配列変数を使うときに、DEFM文でメモリーの増設を行なっているかチェックする。また、同じメモリーを文字変数と数値変数の両方に、同時に使っていないかをチェックする。

●ERR 7：ネスティングエラー
エラーの起きた行がRETURN文やNEXT文であれば、正しくGO-SUB文やFOR文に対応しているかチェックする。また、エラーの起きた行がGOSUB文やFOR文であれば、ネスティングのくり返しをチェックして、GOSUB文は8回まで、FOR文は4回までとする。

●ERR 8：パスワードエラー
パスワードが設定されているときに、別のパスワードを入力しようとしたり、使えないLISTコマンドや、NEW、NEW ALL等の操作をしたかチェックする。

●ERR 9：オプションエラー
カセットインタフェイス<FA-3>やミニプリンタ<FP-12S>が正しく接続されているかチェックする。<FP-12S>は充電されているか、または紙づまりをしていないかチェックする。<FA-3>に接続しているテープレコーダーの音量調整やトーンの調整を変えて再び行なったり、テープレコーダーのヘッドを掃除したり、テープを新しいものと変えてみる。録音のときは白プラグのみを、再生のときは黒プラグのみを差し込んで試してみる。

以上が、エラーに対するチェックポイントです。ここで出てきたコマンドについては、後で順を追って説明していきますが、詳しい説明については103ページからの「コマンドリファレンス」編を参照してください。

(2) エラーは表示されないが、結果が思うように得られない。

このようなデバッグは、プログラム中の計算式や変数の与え方がまちがっている場合が多いので、計算式や変数の動きをチェックする必要があります。
特に計算結果が思うように得られない場合は、元となる公式や計算式と比べてチェックしてください。
プログラムを実行したら止まらなくなったときや、作業をせずに終ってしまうときには、プログラムの流れを制御する変数の動きをチェックしてください。
計算式のチェックは、前例(1)のようにWRTモード(MODE 1と押す)で、計算式の書き込んである行をチェックします。
プログラムの流れをチェックする場合は、制御する変数にデータを入れた後にSTOP文で停止させたり、PRINT文により変数の値を表示させてチェックします。
次のプログラムを記憶させてください。

```
10 INPUT A
20 B=1
30 FOR C=1 TO A
40 B=B*C
50 C=C+1
60 NEXT C
70 PRINT B
80 END
```

このプログラムはINPUT文により入力したデータの階乗を求めるプログラムで、本当は行番号50は不要で、このようにFORループ用の変数を変化させてはいけないのです。
なお、行番号30と60のFOR・NEXT文はくり返し計算を行なわせるためのループ命令です。このFOR・NEXT文については後で説明しますので、ここではとりあえず、記憶させてください。

操　作	表　示
MODE 1	P _1̲23456789
SHIFT P1	P _1̲23456789
NEW EXE └プログラムを消す命令です。	P _1̲23456789
10 SHIFT INPUT A EXE	10 INPUT A
20 B=1 EXE	20 B=1

```
30 SHIFT FOR C=1 SHIFT TO A EXE
40 B=B*C EXE
50 C=C+1 EXE
60 SHIFT NEXT C EXE
70 SHIFT PRINT B EXE
80 END EXE
```

```
30 FOR C=1 T
40 B=B*C
50 C=C+1
60 NEXT C
70 PRINT B
80 END
```

これでプログラムの記憶は完了しましたので、AC MODE 0と押してRUNモードにします。

では実行してみましょう。

操　作	表　示
SHIFT P1	?
例）12 EXE	10395

本当の答は、479001600です。

ここで計算式をチェックしますが、計算式は合っています。次にFOR・NEXTループの流れを見てみます。

行番号50の次にSTOP文を入れて、プログラムを１回ごとに停止させてみます。

操　作	表　示
MODE 1	P _123456789
55 STOP EXE	55 STOP
AC MODE 0	READY P1

このSTOP文は行番号50の次に入れますので、50と60の間を選んで行番号をつけます。ここでは行番号55としました。

では実行してみましょう。

操　作　　　　表　示

SHIFT P1

12 EXE

ここでループの制御変数Cの値を見ます。

C EXE

実行を続けます。

EXE

再び変数Cの値を見ます。

C EXE

この変数Cの値は１から１つづつ増えていかなければいけないのに、２つづつ増えています。これで、FOR・NEXTループの動き（流れ）が正しくないことがわかります。

プログラムをもう一度見なおして、変数Cの流れを見ます。ここでは行番号50に問題があり、必要ないことがわかりますので、行番号50と後から追加した行番号55のSTOP文を削除します。

操　作	表　示
MODE 1	P ☀123456789
50 EXE	_
55 EXE	_
AC MODE 0	READY P1

これでデバッグは完了です。

このSTOP文によるデバッグの他に、トレースモード（MODE 2と押す。"TR"点灯）によるデバッグがあります。

トレースモードでのデバッグは、プログラムを1命令ごとに実行して停止します。この停止している間に変数の値を見たりできますので、EXEキーで1命令ごとに進めてデバッグをします。

前例でためしてください。EXEキーを押すごとにプログラムエリアと行番号を表示します。

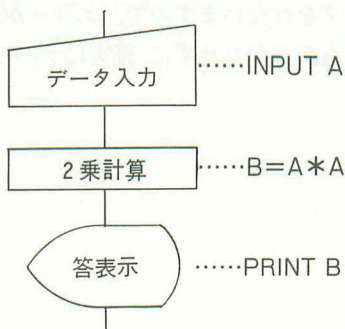
なお、トレースモードを解除するには、MODE 3と押し、"TR"を消します。

以上のようにしてデバッグを行ないますので、エラーが表示されたり、思うように結果が得られなくてもがっかりせずに、確実にデバッグを行なってください。

3−3 発展するプログラム

今迄の説明でプログラムについて、だいたいわかってきたと思います。
プログラムの基本として、入力、計算、出力の3つのポイントがあります。この3つを使うだけでも色々なプログラムが作れますが、ここで説明するコマンドは、憶えておくとプログラムがもっと便利になったり、使いやすくなったりします。ここでは、使って便利なコマンドについて説明していきますので、順に少しづつ憶えていってください。

3−3−1 プログラムの流れを変える〈GOTO文〉

プログラムの3つの基本に加え、同じ計算を何回も繰り返したり、プログラムの流れを行番号の順ではなく、任意の行番号へ移すときに便利なGOTO文があります。
まず例題として、ある数値の2乗を求めるプログラムを作ってみましょう。
このプログラムは「データの入力」、「2乗の計算」、「答の表示」の3つでできますので、フローチャートを作ってみましょう。

このフローチャートにそってプログラムにします。

```
10 INPUT A
20 B=A*A
30 PRINT B
40 END
```

プログラムの記憶方法は、もうわかったと思いますので、ここからは省略します。もしわからなくなったときは、44ページの「プログラムの入力」編をご覧ください。

ここでは例として、15と43の2乗を求めてみましょう。

操　作	表　示
RUN EXE	?
15 EXE	225
RUN EXE	?
45 EXE	1849

このように、1回ごとに実行(RUN EXE)させないとできません。データがいくつもあるような場合には、とても不便です。
この計算が何回も繰り返しできれば便利だと思いませんか？
この繰り返しを便利にする命令がGOTO文です。
GOTO文の働きは、GOTOの後に示される数値に該当する行番号やプログラムエリアに、プログラムの流れを移します。
ここで使っているプログラムにGOTO文を加えてみましょう。それは、行番号40のEND文をGOTO文にします。

```
40 GOTO 10
```

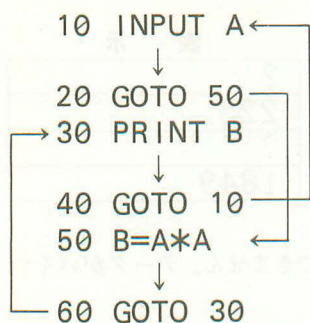
この意味は、以後の流れを行番号10に移します。(ジャンプします)
さっそく、記憶させたプログラムを変更して実行してみましょう。

操　作	表　示
RUN EXE	?
15 EXE	225
EXE	?
43 EXE	1849

このように、GOTO文は繰り返し計算に便利な命令です。
GOTO文はプログラムの先頭に戻して、繰り返し実行させるだけでなく、任意の位置にジャンプすることができますので、他にも便利に使えます。
たとえば、次のプログラムを見てください。

```
10 INPUT A
20 GOTO 50
30 PRINT B
40 GOTO 10
50 B=A*A
60 GOTO 30
```

このプログラムはとてもおかしなプログラムですので、実行させなくてもかまいませんが、使い方は前のプログラムと同じです。

このプログラムの流れは次のようになっています。

このように、GOTO文は指定された行番号に無条件にジャンプしますので、別名"**無条件ジャンプ**"とも呼ばれています。

GOTO文では、行番号へのジャンプの他に、プログラムエリアへのジャンプもできます。GOTOの後に"#"を付けて0～9の1文字でプログラムエリアを指定します。

例) GOTO #1……プログラムエリアP1にジャンプする。
GOTO #9……プログラムエリアP9にジャンプする。

プログラムエリアへのジャンプでは、そのエリア内のプログラムの先頭から実行を続けます。

ワンポイントレッスン

PRINT文

PRINT文は、後に続く変数の内容や文字列、数値を表示する命令です。
変数は数値変数でも文字変数でもよく、変数の内容を表示します。

例) A=123のとき……PRINT A→123
B$="ABC"のとき……PRINT B$→ABC

文字列を" "(ダブルクォーテーション)で囲んで書けばそのまま表示されますので、メッセージとして使うこともできます。

例) PRINT "CASIO" →CASIO

表示させたいものが2つ以上ある場合は、,(カンマ)または;(セミコロン)で区切ることにより、続けて書けます。

例) PRINT A,B,Z$
PRINT "TOTAL=";T

区切りの,と;の違いは、,では1つ目の内容を表示後"STOP"を点灯して止まり、EXEキーを押すことにより次の内容を表示します。区切りが;のときは1つ目を表示後、続けて一行の中に表示します。

例） 次のプログラムで試してください。

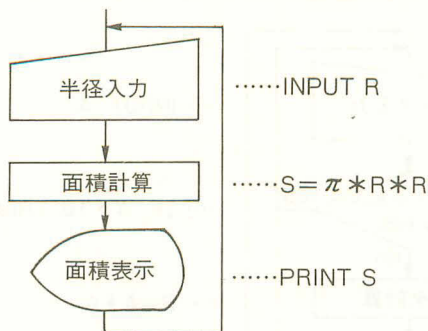
```
10 A=123
20 B$="ABC"
30 PRINT A,B$………Aの内容を表示後、EXEキーを押すことにより
                  B$の内容を表示。
40 PRINT A;B$………AとB$の内容を続けて表示。
50 PRINT B$; ………B$の内容を表示後、停止せずに次に進む。
60 PRINT A   ………Aの内容を表示後、停止する。
70 END
```

このプログラムを実行しますと、次のようになります。

区切りが；のときに続けて表示させても、数値の前に１文字分あいて見えます。これは符号(+,－)を表示しているのですが、正の＋符号は省略されて空白(スペース)として表わされるので、あいているように見えるのです。

〔練習問題〕

任意の半径を入力して、円の面積を求め、GOTO文により繰り返すプログラムを作る。　**公式：S＝πr^2**　（πはSHIFT π と押す）

まず、フローチャートは次のようになります。変数は公式にそってS、Rを使います。

プログラム

```
10 INPUT R              10 INPUT R
20 S=π*R*R     または   20 S=π*R↑2   2乗計算は"↑2"とも書きます。
30 PRINT S              30 PRINT S
40 GOTO 10              40 GOTO 10
```

3−3−2 プログラムに判断させる<IF〜THEN文>

プログラム中で大小の判断や自動的に制御ができたら、とても便利なものです。この判断をプログラム中で行なう命令が**IF文**です。

IF文は"IF"と"THEN"の間の比較式により判断します。

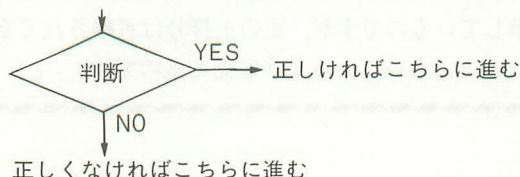
IF 比較式 THEN {行番号またはプログラムエリア / 命令文}

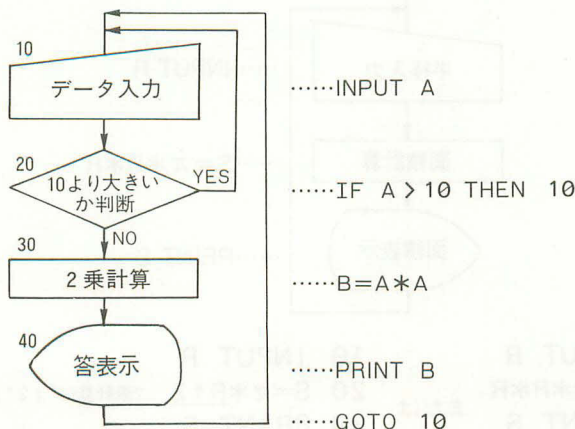
比較式の比較結果により、正しければ"THEN"以降の行番号またはプログラムエリアにジャンプしたり、命令文を実行します。正しくなければ、次の行から実行を続けます。

では実際に、IF文の働きを見てみましょう。

例）任意の数を入力して、10より大きければ再び入力に戻り、10より小さければその値の2乗を求め、答を表示して再び入力に戻る。

このプログラムを分析しますと、「**入力部**」、「**判断部**」、「**計算部**」、「**表示部**」の4つからできています。

判断部のフローチャート記号は次のものを使います。

この記号の"YES"と"NO"の位置は逆でもかまいませんが、わかりやすくするために、"YES"、"NO"を付けておきます。

では、フローチャートにしてみましょう。

フローチャートの左側に付いている数字は行番号で、このようにしておくと、すぐにプログラムにすることができます。

```
10 INPUT A
20 IF A>10 THEN 10
30 B=A*A
40 PRINT B
50 GOTO 10
```

行番号20がIF文の使い方で、比較式の結果が正しければTHEN以降を実行しますので、この場合はTHEN 10で、行番号10へジャンプします。
比較式に用いる比較記号には次のものがあります。

左辺＞右辺……右辺より左辺が大きい
左辺＜右辺……右辺より左辺が小さい
左辺＝右辺……右辺と左辺が等しい
左辺≧右辺……右辺より左辺が大きいか等しい
左辺≦右辺……右辺より左辺が小さいか等しい
左辺≠右辺……右辺と左辺が等しくない

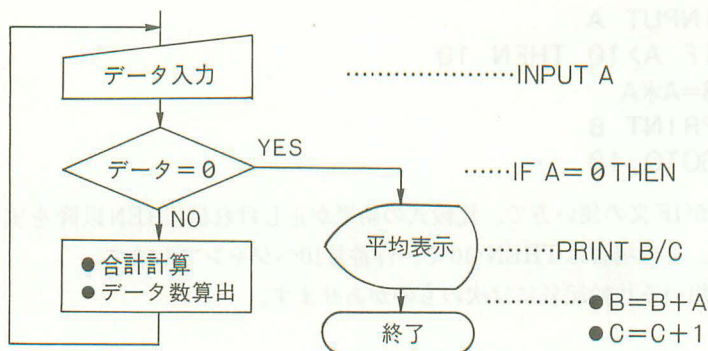
また、〝THEN〟は〝GOTO〟の意味も含まれていますので、〝THEN GOTO 10〟は〝THEN 10〟と書くことができます。
では、このプログラムを実行してみましょう。

操　作	表　示
RUN EXE	?
5 EXE	25
EXE	?
12 EXE	?
9 EXE	81

このように、IF文により判断して、データを選ぶことができます。

例）いくつかのデータを入力して、〝0〟を入力したときにデータの平均を求める。

このプログラムを分析しますと、「入力部」「判断部」「計算部」「表示部」に分けられます。しかし計算部は平均を求めますので、「合計を求める」「個数を数える」「平均を求める」の3つがあります。この3つのうち、平均を求める計算だけは〝0〟を入力したときだけ実行すれば良いので、判断の後に続くと考えられます。

この分析にもとづいて、フローチャートを作ってみます。

このフローチャートからわかるように、データを変数Aに入れ、変数Aが0であるかをIF文により判断し、0でなければ合計とデータ数を求めて、再びデータ入力に戻ります。もしAが0であれば平均を表示させ、終了します。ただし、1回目の入力が0であると0で割り算をすることになるので、エラーとなりますから、注意してください。

この例は前回より少しむずかしくなっていますので、ゆっくりと考えてください。

まず計算部ですが、合計計算は合計を入れておく変数を決め、その変数に入力したデータを加えていく方法がとられます。ここでは合計用の変数をBとしていますので、〝B＝B＋A〟という計算になります。(変数Bの内容に変数Aの内容を加えて、変数Bに記憶させます)データ数は1個のデータにつき1づつカウントしていけばよいので、カウント用の変数(ここではC)に1づつ加えていけばよいのですから、〝C＝C＋1〟となります。

では、肝心な判断部を見ますと、Aが0であれば平均を表示しますので、PRINT文により平均の計算結果を表示させます。PRINT文では変数の値の他に計算式を書けば、計算式の結果(答)を表示しますので、〝PRINT B／C〟となります。IF文ではTHENの後に命令文も書けますので、THENの後に〝PRINT B／C〟を続けて書きます。

ここで1つ問題になることがあります。それは変数Bと変数Cで、このままプログラムにすると、前に何かの値が入っていればどんどん加算されていきますので、答が違ってしまいます。そこで変数Bと変数Cに0を入れておかなければなりません。これは〝B＝0〟〝C＝0〟となります。この2つの代入式に別々の行番号をつけてもかまいませんが、短い文は**マルチステートメント**という〝：〟(コロン)で区切って1行に、〝B＝0：C＝0〟と書くと見易くなります。

では、プログラムにしてみましょう。

```
10 B=0:C=0
20 INPUT A
30 IF A=0 THEN PRINT B/C
40 B=B+A
50 C=C+1
60 GOTO 20
```

これでプログラムができましたが、平均を表示させてもプログラムは終了しません。そこで、行番号30のPRINT文の後にマルチステートメントでEND文を加えます。

```
30 IF A=0 THEN PRINT B/C:END
```

このように、IF文は比較式により判断を行ない、その結果によりプログラムの進行を決めます。

●IF文の応用

今迄出てきた例では、1つの判断によりプログラムの進行を決めていましたが、もし判断が2つ以上あり、全ての条件を満たさなければならないときはどうしましょう。
たとえば、任意の数値を入力し、1～9のものだけを選ぶとします。
これを別の考え方をすれば、0より大きく、10より小さいということですので、〝0＜変数〟と〝変数＜10〟の2つの比較が必要です。
これを1行に書くとすれば次のようになります。

```
IF 0<変数 THEN IF 変数<10 THEN………
```

比較条件が3つ以上の場合も同様にできますが、複雑になりすぎたり、1行が長すぎたりしますので、2つ位にしておくのが良いと思います。

ワンポイントレッスン

マルチステートメント(：)

マルチステートメントは短い代入式などを1行にまとめたり、IF文のTHEN以降でいくつもの命令があるときなどに便利です。

例1)
```
10 A=1  ┐
20 B=2  ├ ⇒ 10 A=1:B=2:C=3
30 C=3  ┘          :
  :
```

例2)
```
50 C=A+B
60 D=A−B
70 E=A*B
80 F=A/B
 :
```
⇒ `50 C=A+B:D=A−B:E=A*B:F=A/B`

IF文中でTHEN以降にマルチステートメントを使う場合は、比較式が正しいときにだけマルチステートメント以降を実行しますので、注意してください。

例3) `IF A<0 THEN A=10:B=20:……`

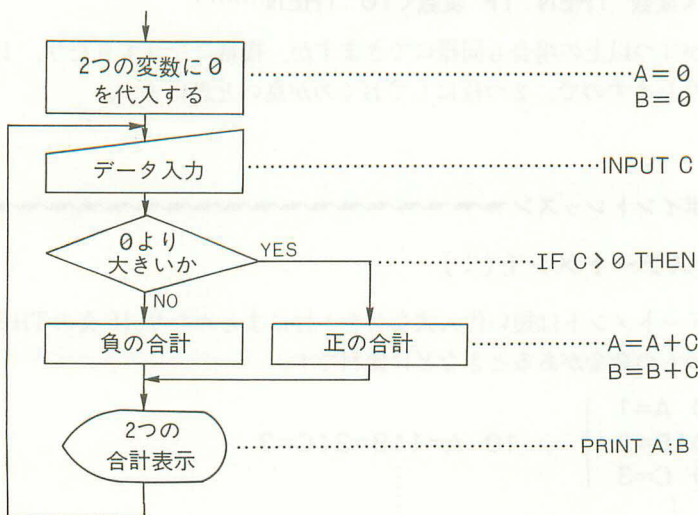
（A=10:B=20:…… は）比較が正しいときにだけ実行します

マルチステートメントはプログラムをまとめるのに便利ですが、あまり1行を長くしすぎると逆に見づらくなりますので、適当な長さで次の行に分けて書いてください。

〔練習問題〕

入力した数値を0より大きい数と小さい数に分けて合計を求めるプログラムを作る。

〈ヒント〉

数値入力後、IF文により2つの変数のどちらかに合計させる。
なお、合計を取る変数はあらかじめ0を代入しておく。

プログラム

```
10 A=0:B=0
20 INPUT C
30 IF C>0 THEN A=A+C:GOTO 50
40 B=B+C
50 PRINT A;B
60 GOTO 20
```

変数AとBに0を代入しておきますが、マルチステートメントで続けなくてもかまいません。
行番号30のIF文では、入力した値(変数Cの値)が0より大きいかを判断して、0より大きいときはTHEN以降で変数Aに合計をとり、そうでないときは行番号40で変数Bに合計をとります。
行番号50では入力するごとに各々の合計を表示させます。

3-3-3 プログラムを繰り返す〈FOR・NEXT文〉

ある一定の回数だけ計算などを繰り返す場合に、GOTO文やIF文の組み合わせではプログラムが長く複雑になってしまいます。繰り返しの回数がわかっているときなどは、もっと簡単にプログラムを組みたいものです。そこで、この繰り返しを簡単に行なう命令がFOR・NEXT文です。
FOR・NEXT文は、FOR文とNEXT文の間にある計算などの作業を指定回数分だけ繰り返します。
FOR文は、

FOR 制御変数=初期値 TO 終値 STEP 刻み幅　　という形式です。

NEXT文は、

NEXT 制御変数　　という形式です。

FOR文中で、制御変数として使えるのはAやBのような1文字の数値変数だけで、配列変数は使えません。初期値、終値、刻み幅は数式や数値変数で与えることができ、初期値から終値を越えるまで、刻み幅づつ制御変数を変化させて、繰り返します。刻み幅は省略でき、省略した場合は1となります。
次のプログラムを記憶させて実行すると、FOR・NEXT文の動きがよくわかります。

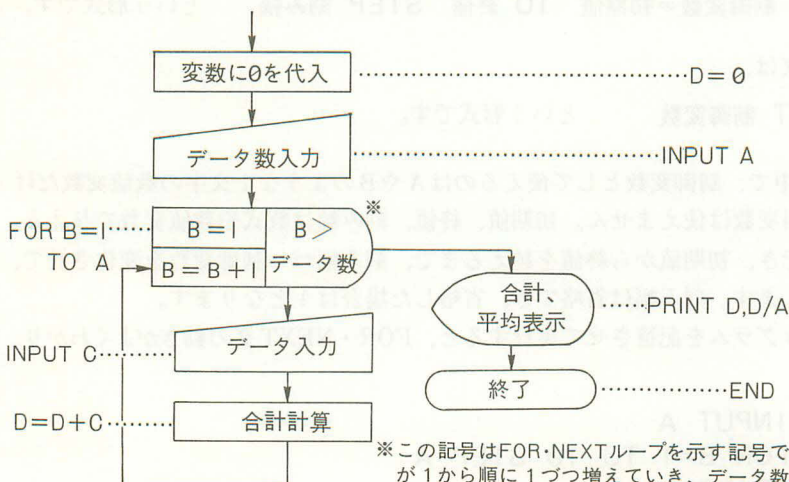
```
10 INPUT A
20 FOR B=1 TO 10 STEP A
30 PRINT B
40 NEXT B
50 GOTO 10
```

このプログラムを実行しますと、次のようになります。

操　作	表　示
RUN EXE	?
3 EXE	1
EXE	4
EXE	7
EXE	10
EXE	?
0.8 EXE	1
EXE	1.8
EXE	2.6
EXE	3.4
⋮	⋮

このように、FOR・NEXT文は初期値から終値までを刻み幅で繰り返します。なお、制御変数に使う変数名は、ここではBを使っていますが、一般にはI、J、Kがよく使われます。

例）データ数を入力し、データの合計と平均を求めるプログラムを作る。

このプログラムでは、まずデータ数を入力し、その後にFOR・NEXT文により個々のデータを入力して合計を求めます。データの入力が終れば、後は合計と平均を表示させます。
最初にフローチャートを作ってみましょう。

※この記号はFOR・NEXTループを示す記号で、変数Bが1から順に1づつ増えていき、データ数より大きくなったらFOR・NEXTループから抜け出て、次の作業に移すことを表わします。

このフローチャートをもとにしてプログラムを作ります。

```
10 D=0
20 INPUT A
30 FOR B=1 TO A
40 INPUT C
50 D=D+C
60 NEXT B
70 PRINT D,D/A
80 END
```

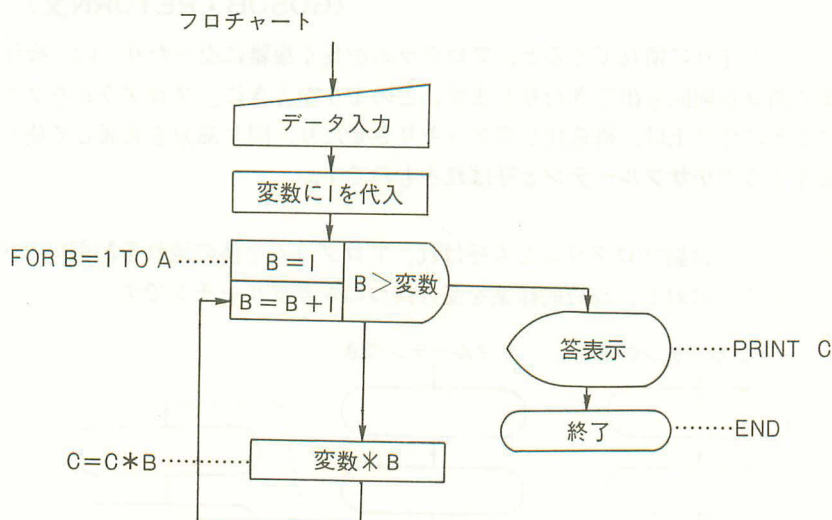
このように、データ数がわかっている場合には、FOR・NEXT文により繰り返してデータ入力や計算を行なうことができます。

行番号20のように、データ数をINPUT文により入力せずに、直接、行番号30のAのかわりに書き込むこともできます。このときは行番号20は不要となります。

〔練習問題〕

階乗を計算するプログラムを作る。

〈ヒント〉

5！＝1×2×3×4×5ですので、FOR・NEXT文により回数分ループさせて計算します。

プログラム

```
10 INPUT A
20 C=1
30 FOR B=1 TO A
40 C=C*B
50 NEXT B
60 PRINT C
70 END
```

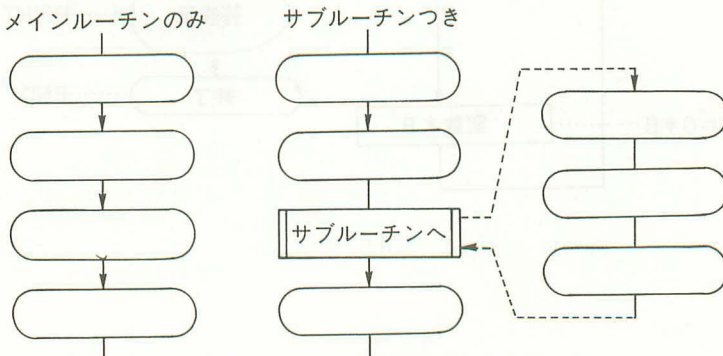
行番号20は階乗を求める変数Cに1を代入しておきます。これは、階乗計算のときにCの値が初期値でないと答がかわってくるからです。

行番号30から50のFOR・NEXT文で階乗計算をします。これは、変数Bの値を1から1つづつ増やしてゆき、1×2×3×……と計算して階乗を求めます。

行番号70のEND文により、1回の計算でプログラムは終了しますが、繰り返し計算するときは、行番号70を〝GOTO 10〟としてください。

3−3−4 複雑なプログラムに便利なサブルーチン 〈GOSUB・RETURN文〉

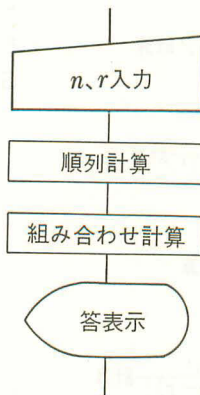
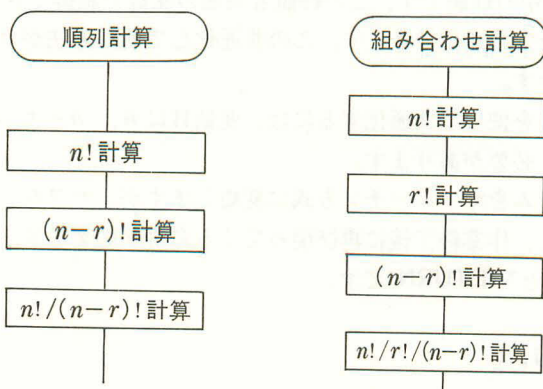
プログラム作りに慣れてくると、プログラムが長く複雑になったり、同じ操作をする部分が何回も出てきたりします。このようなときに、プログラムをブロックごとに作り上げ、構造化してスッキリさせたり、同じ部分を共通して使えるようするのが**サブルーチン**と呼ばれるものです。

サブルーチンは副プログラムとも呼ばれ、プログラム全体の流れを制御するメインルーチンに対し、部分的作業を受け持つのがサブルーチンです。

では、実際のプログラム例で働きを見てみましょう。

例）順列と組み合わせを求めるプログラムを組む

公式： 順　　列　$_nP_r=\frac{n!}{(n-r)!}$

組み合わせ　$_nC_r=\frac{n!}{r!(n-r)!}$

このプログラムでは2つのデータ(n、r)を入力し、順列と組み合わせを求めます。まず、簡単フローチャートを作ってみましょう。

このフローチャートの中で、順列と組み合わせの計算の部分をもう少し詳しくしてみましょう。

2つの計算の中で、$n!$、$(n-r)!$を求める計算が共通していることに気づくと思います。そして、さらに細かく見ると、階乗計算が3回行なわれます。この階乗計算は前節の練習問題で行なっていますので、わからない方は前に戻ってもう一度やりなおしてください。3回の階乗計算をその都度行ないますと、かなり長いプログラムとなります。

プログラム例 (1)

```
 10 INPUT N,R
 20 A=1
 30 FOR B=1 TO N
 40 A=A*B
 50 NEXT B
 60 E=A
 70 A=1
 80 FOR B=1 TO N-R
 90 A=A*B
100 F=A
110 NEXT B
120 P=E/F
130 A=1
140 FOR B=1 TO R
150 A=A*B
160 NEXT B
170 G=A
180 C=E/G/F
190 PRINT P,C
200 END
```


メモリー内容

N：n
R：r
P：順列
C：組み合わせ
A：階乗用変数
B：FOR・NEXT ループ用変数
E：$n!$
F：$(n-r)!$
G：$r!$

このプログラムの中で、3つの階乗計算は共通のプログラムを使っています。異なる点はFOR 文中の終値です。この終値も共通の変数で制御できれば、この3つの計算を完全に共通化できます。この共通化して使う方法がサブルーチンとしての使い方です。

なお、終値に変数Hを使って共通化するには、変数Hにn、$n-r$、r の値を前もって入れておく必要があります。

では、このプログラムをサブルーチン方式に変更しますが、サブルーチンに作業を引き渡す命令と、作業終了後に再び戻ってくる命令が必要です。この2つの命令が"GOSUB"と"RETURN"です。

プログラム例 (2)

```
10 INPUT N,R
20 H=N
30 GOSUB 150
40 E=A
50 H=N-R
60 GOSUB 150
70 F=A
80 P=E/F
90 H=R
```

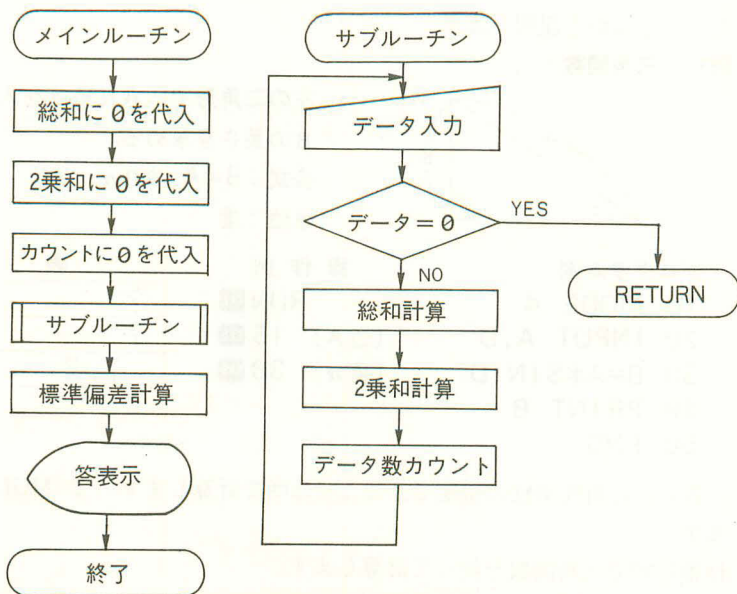
```
100 GOSUB 150
110 G=A
120 C=E/G/F
130 PRINT P,C
140 END
150 A=1                ┐
160 FOR B=1 TO H       │
170 A=A*B              ├ サブルーチン
180 NEXT B             │
190 RETURN             ┘
```

サブルーチンはこのように共通な部分を持つプログラムを、短かく、スッキリと仕上げてくれます。
この例では1行分しか短かくなっていませんが、もっと複雑になってきたり、大きなプログラムを作り上げるうえでは、重要な役割りを持っています。初めのうちはあまり使わないかもしれませんが、覚えておくと便利に使えます。

〔練習問題〕

標準偏差を求めるプログラムを作る。ただし、データ入力部と総和、2乗和、データ数の算出部分はサブルーチンとして作る。

公式： $\sigma n=\sqrt{\dfrac{\Sigma x^2-(\Sigma x)^2/n}{n}}$　　Σx ：総和　　n ：データ数　　Σx^2 ：2乗和

〈ヒント〉　データ入力後、IF文により〝0〞であればメインルーチンに戻り、標準偏差を求める。平方根はSQRを使う。

プログラム

```
 10 B=0:C=0:D=0
 20 GOSUB 100               ……サブルーチンへ
 30 E=SQR((C-B*B/D)/D)……標準偏差
 40 PRINT E
 50 END
100 INPUT A                                  ┐
110 IF A=0 THEN RETURN                       │
120 B=B+A            ……………総和              │
130 C=C+A*A          ……………2乗和             ├サブルーチン
140 D=D+1            ……………データ数          │
150 GOTO 100                                 ┘
```

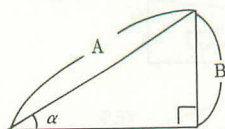
このプログラムでは、行番号20でサブルーチンへ作業を引き渡し、行番号100からのサブルーチンでデータ入力と総和、2乗和、データ数のカウントを求めています。

行番号110のIF文はデータ入力終了の判断で、0を入力すると"THEN"以降に進み、メインルーチンへ戻ります。

なお、行番号50のように、メインルーチンの最後には必ずEND文を入れてください。

3-3-5 関数を使う

マニュアル計算でも使いましたが、プログラム中に関数を書き込んでも使えます。プログラム中でも使い方は同じですので、ここでは実際のプログラム例としていくつかを説明します。

例1） 三角関数

左の三角形で辺Aと角αを入力して、辺Bの長さを求める。

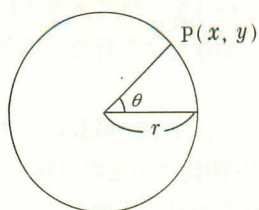
公式：B=A・sin α

単位：度

プログラム例	操作例	表示
10 MODE 4	RUN EXE	?
20 INPUT A,D	（辺A） 15 EXE	?
30 B=A*SIN D	（角α） 30 EXE	7.5
40 PRINT B		
50 END		

行番号10は角度単位の指定で、ここでは度で計算しますので"MODE 4"となります。

行番号30で三角関数を使って計算します。

例2) 三角関数

左の円で、半径rと角度θを入力して点Pの座標(x, y)を求める。

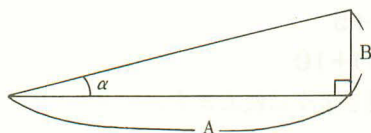
公式：$x = r \cdot \cos\theta$

　　　$y = r \cdot \sin\theta$

単位：ラジアン

プログラム例	操作例	表示
10 MODE 5	RUN EXE	?
20 INPUT R,T	(半径) 5 EXE	?
30 X=R*COS T	(角θ) π/3 EXE	2.5
40 Y=R*SIN T	EXE	4.330127019
50 PRINT X,Y		
60 END		

角度単位がラジアンですので、行番号10でMODE 5を指定します。
行番号30と40でx座標、y座標を求めます。

例3) 逆三角関数

左の三角形より、辺A、Bを入力して角度αを求める。

公式：$\alpha = \tan^{-1}\frac{B}{A}$

単位：度

プログラム例	操作例	表示
10 MODE 4	RUN EXE	?
20 INPUT A,B	(辺A) 100 EXE	?
30 D=ATN(B/A)	20 EXE	11.30993247
40 PRINT D		
50 END		

例4) 10進↔60進変換

時間計算をするプログラムを作る

プログラム例	操作例	表示
10 T=0	RUN EXE	?
20 INPUT D,E,G	(時) 1 EXE	?
30 S=SGN D	(分) 25 EXE	?
40 D=ABS D	(秒) 36 EXE	1°25'36
50 T=T+S*DEG(D,E,G)	EXE	?
60 PRINT DMS$(T)	(時) 2 EXE	?
70 GOTO 20	(分) 15 EXE	?
	(秒) 5 EXE	3°40'41

行番号20では時・分・秒を3つの変数D、E、Gに入力します。
行番号30と40は減算するときのためで、時に負符号(−)をつけて入力すれば、前回までの結果から次の時間を減算します。もし、加算だけを行なう場合は不要です。
行番号50で合計を求めますが、変数Sには加算のときは1が、減算のときは−1が入っていますので、加算・減算ができます。DEG関数は60進数の時・分・秒を10進数に変換する関数で、合計計算は10進数で行なわれます。
行番号60は表示用で、10進数を60進数に変換するDMS$関数を使って表示します。

例5) 乱数発生
1から99までの乱数を作る。

プログラム例

```
10 R=INT(RAN#*99)+1
20 PRINT R
30 GOTO 10
```

行番号10で乱数を作ります。この例では1から99ですので、RAN#に99をかけて、1を加えます。
乱数が5から9のとき→ INT(RAN#*5)+5
乱数が10から20のとき→ INT(RAN#*11)+10
この他にもいくつかの関数がありますが、同じように使えます。

3−3−6 配列を使う

配列とは配列変数のことで、一般のAからZまでの変数の使い方とは異なり、同じ変数名でも番号で管理する変数です。
変数名はアルファベット1文字で、AからZまでが使え、その変数名に管理番号がつきます。

```
例)  A(1)
     │ └──管理番号(添字)
     └────変数名
```

配列変数は、多量のデータを扱うときに便利ですので、使い方を充分覚えてください。
たとえば、10個のデータを入力するには、

```
10 FOR A=1 TO 10
20 INPUT N(A)
30 NEXT A
```

この場合の配列のとり方は、次のようになっています。

通常の変数	O	P	Q	R	S	T	U	V	W	X
配列変数	N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	N(9)	N(10)

50個のデータの中から一番大きいものを選び出すには、

```
10 A=0
20 FOR B=1 TO 50
30 IF P(B)>A THEN A=P(B)
40 NEXT B
50 PRINT A
```

配列変数を使用するときは、配列の取り方に注意してください。配列変数として使う変数と、一般の変数として使う変数の箱が、一部共通に使う所がありたとえば変数名をAとした場合、A(0)はAと同じ箱で、A(1)はBと同じ箱……というように共通の箱を使っていますので、A(0)とAを同じプログラム内で使うと変数内のデータがかわってしまいます。
共通している部分は次のようになっています。

```
 A     B     C     D                X      Y      Z
A(0)  A(1)  A(2)  A(3)..........A(23)  A(24)  A(25)
      B(0)  B(1)  B(2)..........B(22)  B(23)  B(24)
              :                          :
              :                          :
                                X(0)   X(1)   X(2)
                                       Y(0)   Y(1)
                                              Z(0)
```

たとえば、次の操作をしてみてください。

変数Iに7を代入します。

I=7 EXE　　_

変数Iの内容を確認します。

I EXE　　7

配列変数Aの8番目に10を代入します。

A(8)=10 EXE　　_

変数Iの内容を確認します。

I EXE　　10

変数Ｉの内容が変わりました。これは、変数Ｉと配列変数Ａ(８)は同じ箱を使っているからです。
実際に配列変数をプログラム中で使う場合は、FOR・NEXTループや代入用の変数を確保したうえで、配列変数名をさがしてください。

例1) 配列変数を10個必要とする場合
配列変数　A(0)～A(9)
その他の変数　K～Z

例2) 配列変数を50個必要とし、その他の変数は15個で足りる場合
その他の変数　A～O
配列変数　P(0)～P(49)〔DEFM 39 EXE が必要です〕

例3) FOR・NEXTループ用に変数Ａを、合計用に変数Ｂを使用し、配列変数を100個必要とする場合

```
10 DEFM 76             ………メモリーを76個増設して102個とする
20 B=0
30 FOR A=0 TO 99  ┐
40 INPUT C(A)     │……配列はC(0)～C(99)が使えます
50 B=B+C(A)       │
60 NEXT A         ┘
```

このように、配列変数を使って多量のデータを扱う場合は、その他の必要な変数と重ならないように注意してください。
なお、配列変数で使用した場合と通常のＡ～Ｚの変数として使用した場合との関係については173ページの表を参照してください。

配列変数を使用するときのもう一つの注意点は、**メモリーの増設**です。一般変数で使うＡからＺと共通する部分を使う場合は、増設する必要ありませんが、それ以上の変数を使う場合は必ず増設を行なってください。
メモリーの増設は**DEFM命令**により行ないます。
DEFM命令は、１個単位でいくつ増設するかを指定します。
たとえば、10個を増設する場合、

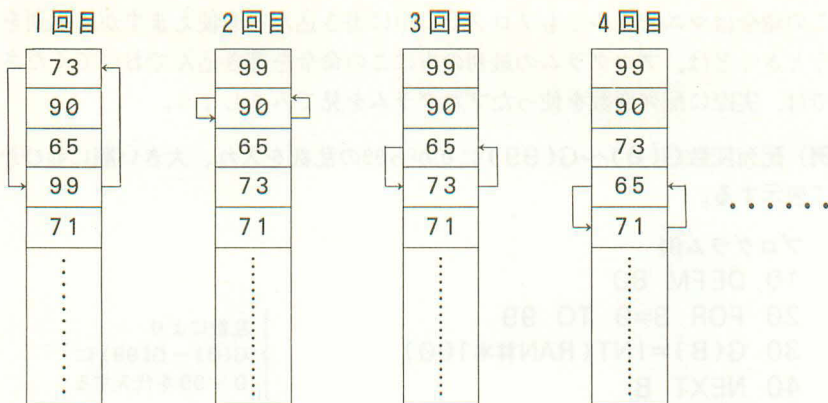
マニュアルでは、DEFM 10 EXE
プログラム中では、行番号DEFM 10

この命令はマニュアルでもプログラム中に書き込んでも使えますが、配列を使うときなどは、プログラムの最初の方にこの命令を書き込んでおいてください。
では、実際に配列変数を使ったプログラムを見てみましょう。

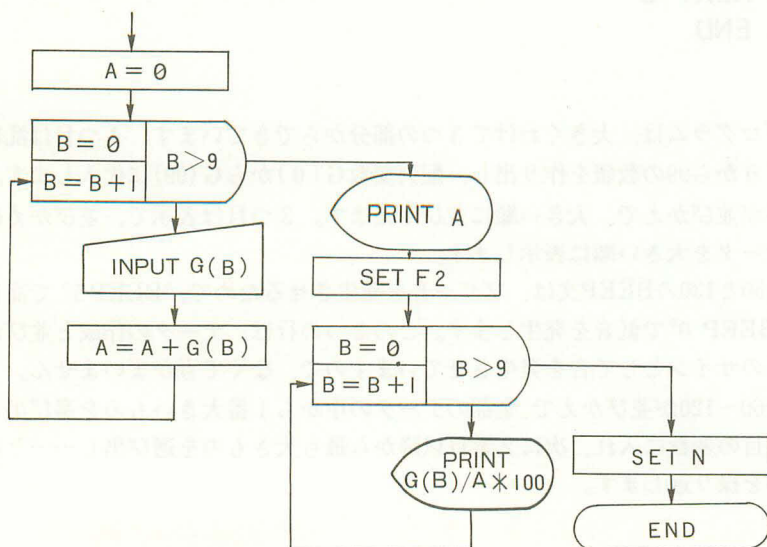
例）配列変数G(0)〜G(99)に0から99の乱数を入れ、大きい順に並びかえて表示する。

プログラム例

```
 10 DEFM 80
 20 FOR B=0 TO 99                       } 乱数により
 30 G(B)=INT(RAN#*100)                  } G(0)～G(99)に
 40 NEXT B                              } 0～99を代入する
 50 BEEP 1
 60 FOR B=0 TO 99                       }
 70 A=-1                                }
 80 FOR C=B TO 99                       }
 90 IF G(C)>A THEN A=G(C):D=C           } 大きい順に並びかえる
100 NEXT C                              }
110 E=G(B):G(B)=G(D):G(D)=E             }
120 NEXT B                              }
130 BEEP 0
140 FOR B=0 TO 99                       }
150 PRINT G(B)                          } 順番に表示する。
160 NEXT B                              }
170 END
```

このプログラムは、大きくわけて3つの部分からできています。1つ目は乱数により0から99の数値を作り出し、配列変数G(0)からG(99)に代入します。2つ目が並びかえで、大きい順に並びかえます。3つ目は表示で、並びかえ終ったデータを大きい順に表示します。
行番号50と130のBEEP文は、ブザー音を発生させるためで、〝BEEP 1〞で高音を、〝BEEP 0〞で低音を発生します。この2つの行は、データの作成と並びかえ終了のサインとして音を発生させていますので、なくてもかまいません。
行番号60〜120が並びかえで、全部のデータの中から1番大きいものを選び出して1番目の変数に入れ、次に2番目以降から最も大きものを選び出し……という操作を操り返します。

なお、行番号10のDEFM文はメモリーの増設で、G(0)~G(99)までの100個の変数の内、20個がG~Zの変数となりますので、残りの80個を増設します。このように配列変数を用いると、多量のデータを扱うのに便利になります。

〔練習問題〕

10個のデータを入力し、合計と各々のデータの構成比を求める。構成比は百分率で、小数以下2桁まで求める。

プログラム

```
 10 A=0
 20 FOR B=0 TO 9
 30 INPUT G(B)
 40 A=A+G(B)
 50 NEXT B
 60 PRINT A
 70 SET F2
 80 FOR B=0 TO 9
 90 PRINT G(B)/A*100
100 NEXT B
110 SET N
120 END
```

行番号20～50までがデータ入力部で、FOR･NEXT文によりG(0)～G(9)に10個のデータを入力します。行番号70の〝SET F2〞は、構成比を百分率で小数以下2桁まで表示するための小数以下指定です。行番号80～100で構成比を百分率で表示します。行番号110では、小数以下指定を解除しておきます。

3-3-7 データを読み込む<READ・DATA・RESTORE文>

ここでは、データ処理の一つの方法として、プログラム中にデータを書き込む〝READ〞、〝DATA〞、〝RESTORE〞について説明します。
データの入力方法にINPUT文がありますが、INPUT文はプログラム実行中に〝?〞を表示してキーボードからデータを入力します。READ文はプログラム中のDATA文に書かれているデータを変数に読み込みます。

例）DATA文にある10個のデータを読み込み、表示する。

プログラム

```
10 FOR B=1 TO 10
20 READ A
30 PRINT A
40 NEXT B
50 DATA 1,2,3,4,5,
   6,7,8,9,10
60 END
```

操作	表示
RUN EXE	1
EXE	2
EXE	3
⋮	⋮
EXE	10

READ文は、必ずDATA文と対で使われ、"READ"に続く変数に、DATA文から持ってきたデータを代入します。
READ文の後の変数は ,(カンマ)で区切ることにより、いくつでもかくことができます。

例）プログラム　　操作　　表示

```
10 READ A$,B$
20 PRINT A$;B$
30 END
40 DATA CASIO,PB&FX
```

操作	表示
RUN EXE	CASIOPB&FX

DATA文の位置はプログラム中のどこにあってもかまいません。プログラム実行後は行番号の若いDATA文の最初のデータから順に変数に代入していきます。
READ文中の変数は、DATA文に書いてあるデータが数値のときは数値変数を、文字のときは文字変数を使ってください。

例）プログラム　　操作　　表示

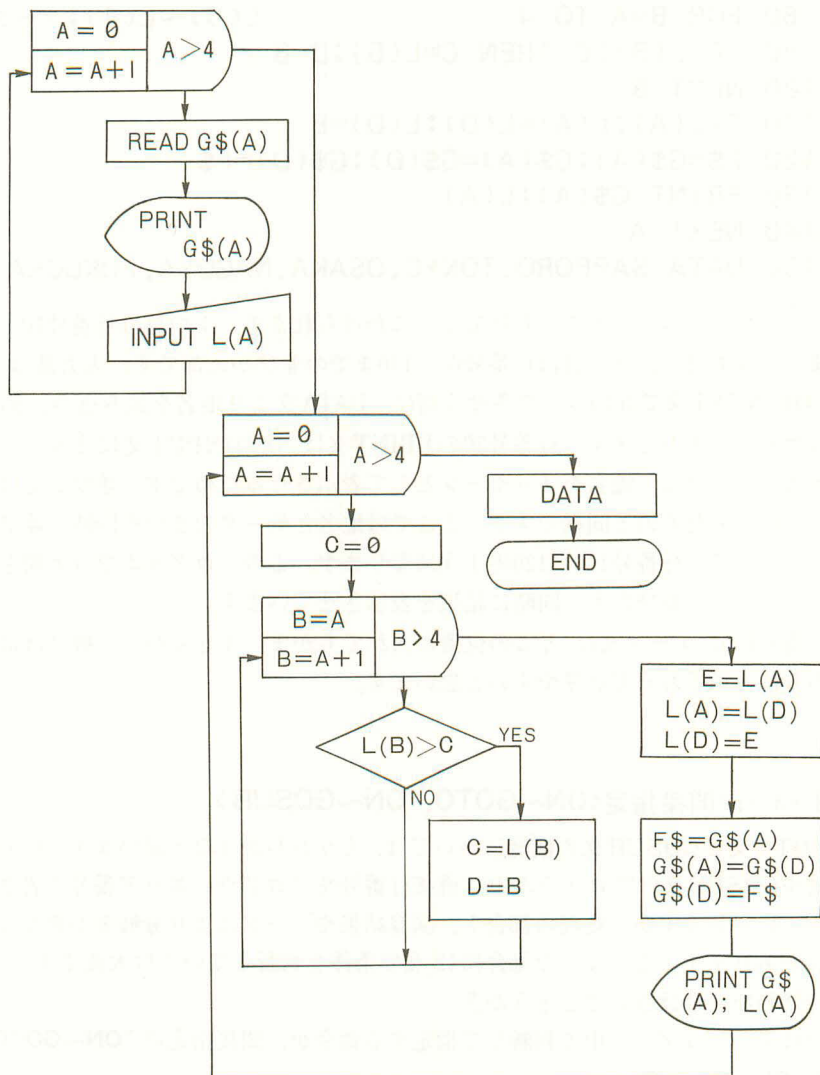
```
 10 RESTORE
 20 READ A$
 30 PRINT A$
 40 READ B
 50 PRINT B
 60 RESTORE 90
 70 READ C$
 80 PRINT C$
 90 DATA ABC
100 DATA 123456
110 END
```

操作	表示
RUN EXE	ABC
EXE	123456
EXE	ABC

RESTORE文で、行番号を指定しないとき、次のREAD文で読み込むデータは、そのプログラム中に書かれている最初のDATA文の最初のデータです。行番号の指定があるとき、次のREAD文で読み込むデータは、指定した行番号のDATA文の最初のデータです。

〔練習問題〕

札幌、東京、名古屋、大阪、福岡の地名をプログラム中で読み込み、付随するデータを入力して、大きい順に並びかえる。
ただし、地名はG$(0)～G$(4)に読み込み、データはL(0)～L(4)に入力する。

フロチャート例

プログラム例

```
 10 FOR A=0 TO 4
 20 READ G$(A)
 30 PRINT G$(A);
 40 INPUT L(A)
 50 NEXT A
 60 FOR A=0 TO 4
 70 C=0
 80 FOR B=A TO 4
 90 IF L(B)>C THEN C=L(B):D=B
100 NEXT B
110 E=L(A):L(A)=L(D):L(D)=E
120 F$=G$(A):G$(A)=G$(D):G$(D)=F$
130 PRINT G$(A);L(A)
140 NEXT A
150 DATA SAPPORO,TOKYO,OSAKA,NAGOYA,FUKUOKA
```

メモリー

A :FOR・NEXT文用
B :FOR・NEXT文用
C :最大値
D :最大値の番号
E :並びかえ用
F$:並びかえ用
G$(0)〜G$(4):地名
L(0)〜L(4):データ

このプログラムは、大きくわけて2つにわけられます。1つ目は行番号10〜50までの入力部で、2つ目は行番号60〜140までの並びかえ部です。入力部はFOR・NEXT文で5回ループさせる間に、DATA文より地名を読み込み、同時にデータも入力します。行番号30のPRINT文は、次のINPUT文によりデータ入力をする前に、地名をメッセージとして表示させるためです。並びかえは前回に行なったものと同様ですが、ここでは地名とデータの2つを同時に並びかえますので、行番号110と120のようになります。このプログラムでは手間をはぶくために、並びかえと同時に結果を表示させています。

行番号150のデータ文は、どこの位置に入れてもかまいませんが、一般には最後の方に入れておく方が見やすいと思います。

3−3−8 間接指定<ON〜GOTO、ON〜GOSUB>

GOTO文とGOSUB文の機能については、もうおわかりだと思います。GOTO文やGOSUB文はプログラム中に直接行番号やプログラムエリア番号を書き込んで指定しますが、処理の都合上、演算結果やデータにより分岐先が異なる場合があります。このような場合にIF文で条件を判断していては大変です。もっと便利な命令はないでしょうか。?

分岐先をプログラム中で判断して指定する命令が、間接指定の"ON〜GOTO"と"ON〜GOSUB"です。

ON〜GOTO文とON〜GOSUB文は似たような機能を持っています。

例）

"ON"に続く数値変数や計算式の結果により、1であれば1つ目の分岐先に、2であれば2つ目の分岐先に、……と分岐します。変数や計算式の値よりも分岐先の数が少ないときや該当する分岐先がないときは、次の行またはマルチステートメント以降の命令に進みます。

例）

```
 10 INPUT A
 20 ON A GOTO 100,200,300,400,500
 30 PRINT " NO"
 40 END
100 PRINT "LINE 100" :END
200 PRINT "LINE 200" :END
300 PRINT "LINE 300" :END
400 PRINT "LINE 400" :END
500 PRINT "LINE 500" :END
```

〔練習問題〕

角度と4～6の数値を入力し、間接指定で角度単位を指定するサブルーチンへ分岐して、サイン(SIN)の答を求める。

フロチャート

プログラム

```
 10 INPUT "KAKUDO",A
 20 INPUT "TANI",B
 30 ON B-3 GOSUB 100,200,300
 40 PRINT SIN A
 50 GOTO 10
100 MODE 4
110 RETURN
200 MODE 5
210 RETURN
300 MODE 6
310 RETURN
```

このプログラムは2つのデータを入力しますので、INPUT文にメッセージを加えて入力しやすくしました。行番号10では角度を変数Aに入力し、行番号20では角度単位として4、5、6のいずれかを変数Bに入力します。行番号30ではON~GOSUB文を使い、4~6の数値を1~3に直して分岐先を指定します。サブルーチンでは各々角度単位を指定して、元に戻ります。

3-3-9 文字を扱う文字関数〈LEN、MID$、VAL、STR$〉

SINやCOSなどは数値を扱う関数ということで、数値関数と呼ばれていますが、この他にも文字を扱う文字関数があります。本機には文字関数として"LEN" "MID$"、"VAL"、"STR$"があります。

●LEN

LEN関数は文字変数内の文字列の数を数えます。

例) [A][SHIFT][$][=][SHIFT]["][A][B][C][SHIFT]["][EXE]
[L][E][N][SHIFT][(][A][SHIFT][$][SHIFT][)][EXE]

表示
3

※ LEN関数では配列変数は使えません。

●MID$

MID$関数は専用文字変数($)に記憶されている文字列の中から、いくつかを取り出す関数です。取り出し方は、"どこから"、"いくつ"取り出すかを指定します。

例)
```
10 $="CASIO PB&FX"
20 PRINT $
30 PRINT MID$(1,5)
40 PRINT MID$(7,5)
50 END
```

操作	表示
RUN [EXE]	CASIO PB&FX
[EXE]	CASIO
[EXE]	PB&FX

●VAL

VAL関数は文字変数内に記憶されている**数字**を**数値**に変換します。

例)
```
10 A$="123":B$="456"
20 PRINT A$+B$
30 PRINT VAL(A$)+VAL(B$)
40 END
```

操　作	表　示
RUN EXE	123456
EXE	579

※ VAL関数では配列変数は使えません。

●STR$

STR$関数はVAL関数とは逆に、数値変数内に記憶されている**数値**を**数字**に変換します。

例)
```
10 A=123:B=456
20 PRINT A+B
30 PRINT STR$(A)+STR$(B)
40 END
```

操　作	表　示
RUN EXE	579
EXE	123456

例)
```
10 INPUT $
20 FOR A=1 TO LEN($)
30 PRINT MID$(A,1);
40 BEEP 1
50 NEXT A
60 END
```

このプログラムは、専用文字変数に入力された文字を、MID$関数により1文字づつ取り出します。取り出す位置の指定はFOR・NEXT文により決めますが、終値はLEN関数により求めます。
行番号30の最後の";"は、続けて表示させるため、表示しても止まらないようにします。

例)
```
 10 A=1:B=0
 20 PRINT "<";STR$(A);">";
 30 INPUT $
 40 IF $="END" THEN 100
 50 B=B+VAL($)
 60 A=A+1
 70 GOTO 20
100 PRINT B/(A-1)
110 END
```

このプログラムは、データ数のわからないデータの平均を求めます。データ入力の終了は"END"を入力することにより行なわれ、行番号100に分岐して平均を表示します。
行番号20は入力をしやすくするためのメッセージで、数値変数Aをそのまま表示しますと符号桁として数字の左側が１文字分あきますので、文字としてあかないようにしています。
行番号50では、データを文字として専用文字変数に入力していますので、数値に変換して合計計算をします。
なお、このプログラムでは"END"以外の文字を入力しても、データ入力終了とはなりませんので、"END"と数値以外の入力ではエラー(ERR 2)となります。

3−3−10 覚えておくと便利な入出力制御関数〈KEY$、CSR〉

入出力制御関数とは、データの入力や出力を制御する関数で、"KEY$"と"CSR"があります。
KEY$関数は、データを入力するときに使いますが、INPUT文とは次の点が異なります。

INPUT文	KEY$関数
●数値は仮数12桁、指数２桁以内。 ●文字は文字変数７文字、専用文字変数30文字以内。	●押されたキーを１文字分だけ指定された文字変数に読み込みます。
●"？"を表示して入力待ちとなる。	●何も表示せずに入力待ちとならない。

例)
```
10 INPUT A
20 PRINT A
30 B$=KEY$
40 IF B$="" THEN 30
50 PRINT B$
60 END
```

行番号10はINPUT文の使い方ですが、行番号30と40がKEY$関数の使い方です。
KEY$関数はキーボードから１文字(１キー)分の入力だけを受けつけますが、キーを押しても押さなくても入力待ちで停止しません。そのため、行番号40のようにIF文と組み合わせて、キー入力がなければ再び行番号30に戻り、キー入力可能な状態にします。
KEY$関数はこのようにIF文と組み合わせて使われます。

例）

```
 10 A$=KEY$
 20 IF A$="1" THEN 100
 30 IF A$="2" THEN 200
 40 IF A$="3" THEN 300
 50 GOTO 10
100 PRINT "LINE 100":END
200 PRINT "LINE 200":END
300 PRINT "LINE 300":END
```

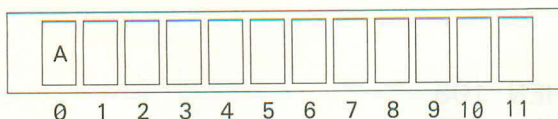
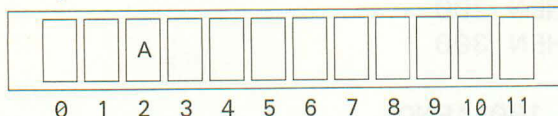
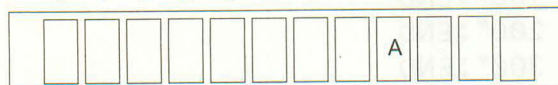
KEY$関数の使い方として、このような例もあります。前例ではキーが押されたかをチェックしていましたが、この例では押されたキーが1か2か3かをチェックして、合っていれば次の作業に進みます。
ただし、この例のようにKEY$関数がプログラムの先頭の方に書き込まれているときは、プログラムのスタートの仕方に注意してください。プログラムのスタート方法として2つの方法がありますが、SHIFT P1のようなスタートの方法では、最後のキー操作として1キーを押しています。プログラムがスタートするとすぐにKEY$関数でキー入力を読み込みますので、数字キーを押したままでは、そのキーが入力されてしまいます。
ためしに、プログラムエリアP1にこのプログラムを記憶させ、SHIFT P1として見てください。1キーを少しでも長く押していると"LINE 100"が表示されます。
もし、先頭でKEY$関数を使う場合は、次のリストを加えてください。

```
 5 A$=KEY$               ┐
 6 IF A$≠"" THEN 5       ├キーがはなされるまで待つ
10 A$=KEY$               ┘
   :
```

CSR関数は、データを表示するときの位置を指定するもので、PRINT文中で使います。

例）

```
10 PRINT "A"
20 PRINT CSR 2;"A"
30 PRINT CSR 8;"A"
40 END
```

この例を実行させますと、CSR関数の働きがわかると思います。
表示上の指定は左端から0、1、2、……11となっていますので、最初のCSR関数なしでは左端から表示されますが、CSR関数で指定しますと指定した位置から表示されます。

例)
```
10 A=INT(RAN#*12)
20 PRINT CSR A;"↑"
30 GOTO 10
```

このプログラムは乱数により0～11の数値を作り出し、CSR関数で"↑"を表示させます。実行後、EXEキーを押すたびに"↑"が色々な所に表示されます。
KEY$関数やCSR関数を色々と組み合わせることにより、おもしろいゲームなどができます。

例)
```
 10 D=0:$=" 0123456789"
 20 FOR B=1 TO 10
 30 IF KEY$≠"" THEN 30
 40 A=INT(RAN#*10)
 50 PRINT MID$(1,A+1);"↑";MID$(A+3);
 60 FOR C=1 TO 30
 70 E$=KEY$
 80 IF E$≠"" THEN 100
 90 NEXT C
100 IF E$<"0" THEN 130
110 IF E$>"9" THEN 130
120 IF VAL(E$)=A THEN D=D+1:BEEP 1:GOTO 140
130 BEEP 0
140 PRINT
150 NEXT B
160 PRINT CSR 2;"RIGHT";D;
170 IF D≠10 THEN 210
180 FOR B=1 TO 10
190 BEEP 1:BEEP 0
200 NEXT B
210 END
```

このプログラムは0から9の数字が並んでいる中で、1文字だけ数字ではなく"↑"が出現します。この出現している場所の数字キー(⓪～⑨)を押すゲームです。

行番号40では乱数により作り出した0～9の数値に対応する場所に"↑"を表示させています。

行番号30でのKEY$の使い方は、キーを1度はなしてから表示させるためで、前のキーを押し続けますと表示は出ません。KEY$関数はその時点のキー入力を読み込みますので、このような使い方もできます。

行番号60以降はキー入力の判断で、キーが押されれば繰り返しから抜け出し、正解かどうかをチェックします。行番号100と110は入力のチェックで、IF文により文字も比較できますので、ここでは0～9以外は行番号130に分岐させてまちがいとします。この文字の比較は170ページのキャラクター表により決っていますので、くわしくは表を参照してください。

行番号120は正解かどうかのチェックで、KEY$では文字として読み込まれますので、VAL関数により数値に変換して比較します。

行番号30のようにキーを押しているかいないかの判断だけでは、文字変数に代入せずに判断してもかまいませんが、行番号70から120までのように何回も判断させて正解を取り出す場合は、一度文字変数に入れて使います。では、このプログラムを記憶させて、遊んでみてください。

操作例

表示のかわる速度が速すぎる方は、行番号60のFOR文の終値を30より大きくすれば、ゆっくりとキーを押せます。また、プログラムの先頭にレベルを入力するINPUT文を追加して、速度を変えてみるのもおもしろいでしょう。

3−4 あると便利なオプション

本機は本体だけでも、かなり便利に使えますが、オプションとしてカセットテープレコーダー用インタフェイス(アダプター)〈FA-3〉とミニプリンタ〈FP-12S〉があります。
〈FA-3〉は本機で組んだプログラムを短時間のうちに記録したり、呼び戻したりできますし、メモリー内のデータも記録しておくことができます。
〈FP-12S〉はプログラムやデータを印字するミニプリンタで、プログラムのリスト(内容)を印字して保存したり、計算結果を印字することができます。
では、各々の機能を簡単に説明していきます。

3−4−1 プログラムやデータを保存する

プログラムやデータをカセットテープに保存するには、〈FA-3〉が必要です。
FA-3とテープレコーダーの接続の仕方および注意事項については、FA-3に付属の取扱説明書をご覧ください。ここでは主に、使い方について説明します。

■プログラムの記録および呼び戻し

本機にプログラムを記憶させていくと、次のプログラムを記憶させたいが、もう入らなくなることがあります。こんなときに、前のプログラムを消してしまっては、後でもう一度使いたいときに不便です。また、RAMカードの電池を交換するときも、プログラムやデータは消えてしまいます。ここでFA-3が真価を発揮します。
プログラムを記録する命令は〝SAVE〞または〝SAVE ALL〞で、〝SAVE〞はＰ0だけとか、Ｐ9だけというように、１つのプログラムエリア内のプログラムだけを記録します。〝SAVE ALL〞はＰ0からＰ9までの10個のプログラムエリア内のプログラムを同時に記録します。

SAVE命令

MODE ∅　　READY Pn

└→この番号のプログラムエリア内のプログラムが記録されます。

SAVE ALL命令

プログラムエリアに関係なく、Ｐ0からＰ9までの10個のプログラムエリア内のプログラムが記録されます。

SAVE命令とSAVE ALL命令は、単独のエリアだけのプログラムであるか、メインプログラムの他にもサブルーチンなどとしてプログラムを使っているかによって使い分けます。
SAVE、SAVE ALL命令ともマニュアルで実行します。

例） SAVE EXE
SAVE "CASIO" EXE
SAVE ALL EXE
SAVE ALL "PB" EXE

SAVEとSAVE ALLの後に続く" "でかこんだ文字は、ファイル名と呼ばれるキーワードで、記録するプログラムに個別の名前をつけておけば、後で呼び戻すときに名前を指定して呼び戻せます。ファイル名は8文字以内のアルファベットや数字などを" "でかこんで書きます。
プログラムの呼び戻しには"LOAD"または"LOAD ALL"を使います。LOAD命令とLOAD ALL命令は、記録したときにSAVEまたはSAVE ALLで記録したかにより使い分けます。

記録＼呼び戻し	LOAD	LOAD "ファイル名"	LOAD ALL	LOAD ALL "ファイル名"
SAVE	○	×	×	×
SAVE "ファイル名"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "ファイル名"	×	×	○	○

※○印は呼び戻し可。
※ファイル名は同一のものとする。

例） LOAD EXE
LOAD "ファイル名" EXE
LOAD ALL EXE
LOAD ALL "ファイル名" EXE

LOAD命令とLOAD ALL命令により、プログラムを呼び戻すとき、プログラムの記録の仕方により次の表示が出ます。

記録方式	表示
SAVE	PF：
SAVE "ファイル名"	PF：ファイル名
SAVE ALL	AF：
SAVE ALL "ファイル名"	AF：ファイル名

SAVE命令により記録したプログラムをLOAD命令により呼び戻すとき、記録したプログラムエリアと呼び戻すプログラムエリアは同一でなくてもかまいません。

例)　P0のプログラムを記録
　　　↓
　　P9に呼び戻す

<ご注意>

プログラムを記録したり、呼び戻すときにうまくいかないことがあります。そのようなときには次の点をチェックしてください。

●記録しているときに"ERR 9"が表示される。

〔チェックポイント〕

本体とFA-3が正しく接続されているかチェックする。

●呼び戻しているときに"ERR 9"が表示される。

〔チェックポイント〕

テープの保存状態が悪く、伸びているようなときは、新しいテープと交換する。

テープレコーダーのヘッドがよごれているときは、市販のクリーニングキットでヘッドをクリーニングする。

テープレコーダーのトーン調整を中位にする。

●呼び戻しているときに、エラーも何も表示されずに、戻ってこない。

〔チェックポイント〕

テープレコーダーの出力ボリウムが低いので、最高(MAX)に近い位置までボリウムを上げる。

テープレコーダーの出力規格がFA-3に合わない。

■データバンクデータの記録および呼び戻し

データバンクに記憶させたデータを他機に移したいときや、RAMカードの電池を交換するときには、データをテープに記録しておきたいものです。

では、データバンクデータをテープに記録してみましょう。

データバンクデータの記録には"SAVE#"を使います。

SAVE#命令は1度に全部のデータを記録できます。

SAVE#　"ファイル名"
　　　　　8文字以内の文字

ファイル名はプログラムの記録と同じで、8文字以内の文字を" "でかこみます。

例）SAVE＃ "MEMO" EXE

テープに記録されたデータバンクデータを呼び戻すには、"LOAD＃"を使います。LOAD＃命令はテープ上に記録されたデータをそのまま呼び戻し、データバンクに記憶させますので、何かが記憶されていても前のデータを消して、新たなデータを記憶します。

LOAD＃ "ファイル名"
8文字以内の文字

例）LOAD＃ "MEMO" EXE

データバンクデータを呼び戻しているとき、次のような表示が出ます。

記録方式	表　示
SAVE＃	MF：
SAVE＃ "ファイル名"	MF：ファイル名

■データの記録、呼び戻し

プログラムにはデータがつきものですが、いちいちキーボードから入力していてはめんどうなものです。
ここでは、メモリー内のデータをテープに記録して、再び呼び戻す方法を行なってみましょう。
データを記録するには"PUT"を使います。
PUT命令はファイル名をつけて、変数名で指定します。

PUT "ファイル名"　変数1，変数2
8文字以内の文字

ファイル名はプログラムの記録と同じで、8文字以内の文字を" "でかこみます。変数名の指定は、専用文字変数（＄）があるときは最初に、次の2つの変数名で、"どこから"、"どこまで"の変数を記録するか指定します。

例）
専用文字変数（＄）とAからMまでの13個の変数の内容を記録する。

PUT ＄,A,M

AからZ(10)までの36個の変数の内容を、ファイル名"CASIO"で記録する。

PUT "CASIO" A,Z(10)

※メモリーが増設してある場合。

変数名の指定はどこから、どこまでのように指定しますので、〝A,Z″という指定となり、〝Z,A″のような指定はできません。
なお、変数を文字変数として使っている場合でも、〝A$, Z$″とせずに〝A, Z″とすることもできます。
データの呼び戻しには〝GET″を使います。
GET命令もファイル名をつけて、変数名で指定します。

GET 〝ファイル名″ 変数1, 変数2
8文字以内の文字

例)
専用文字変数($)とXからZまでの3個の変数にデータを呼び戻す。

```
GET $,X,Z
```

G(0)からG(99)までの変数に、ファイル名〝PB″のデータを呼び戻す。

```
GET "PB" G(0),G(99)
```

※メモリーが増設してある場合。

GET命令により、データを呼び戻しているとき、次のような表示が出ます。

記録方式	表示
PUT $, A, Z	DF:
PUT 〝ファイル名″ G, P	DF:ファイル名

3-4-2 記録を残す

プログラムを作るときに、プログラム内容を印字できれば、デバッグや内容変更にとても便利ですし、演算結果を印字して残すのも便利なものです。
ここではミニプリンタ<FP-12S>を使って印字させてみましょう。
印字させるにはプリントモード(〝PRT″点灯)でキー操作を行ないます。
プリントモードはMODE 7と押すことにより指定され、MODE 8と押すことにより解除されます。
プログラムの内容を印字させるには、MODE 7と押した後、LIST命令を実行します。

例)
次のプログラムを記録させます。

```
10 INPUT A
20 PRINT A*A
30 GOTO 10
```

記憶させた後に次の操作を行ないます。

MODE 7 LIST EXE　　印字例

```
LIST
  10 INPUT A
  20 PRINT A*A
  30 GOTO 10
```

もし、P0からP9までの10個のプログラムエリア全部の内容を印字させたいときは、

LIST ALL EXE

と操作します。

なお、印字後はMODE 8と操作してプリントモードを解除してください。

演算結果等を印字させるには、MODE 7と押すか、プログラム中に"MODE 7"を書き込みます。MODE 7と押しますと、その後のキー操作全てが印字されますので、必要な所だけを印字させたいときは、プログラム中に書き込んだ方が便利です。

※プログラム中に書き込む場合は、MODEキーではなく、M O D Eと押して入力します。

例）

演算結果だけを印字する。

```
10 INPUT A
20 MODE 7
30 PRINT A*A
40 MODE 8
50 GOTO 10
```

このプログラムでは、データ入力後にプリントモードを指定し、印字(表示もします）後プリントモードを解除して、再び入力に戻ります。

なお、プリントモード中ではPRINT文による表示は停止せずに、印字後次の命令に進みます。

MODE 7により印字後は、MODE 8でプリントモードを解除してください。

> ミニキャラクタープリンタは本機専用<FP-12S>をご使用ください。なお、FP-12をお持ちの方は、FA-3を介して接続すれば、そのままご使用になれますが、直接FP-12と本機を接続するには専用の固定台が必要となります。この固定台をご希望の方は、お近くの営業所に「FP-12S用固定台<SB-1>」と指定してご用命ください。

3−5 PB-100のプログラムを使う

PB-100やPB-300で作られたプログラムは、ビジネスをはじめゲームまでたくさんあり、ライブラリーも出版されています。
本機もシリーズの1つとして、この豊富なプログラムを利用できますが、本機にはPB-100/300以上のコマンドを持っていますので、もっと便利な使い方もできます。ここでは、PB-100/300により作られたプログラムを、本機で使ってみましょう。

■相異点

本機とPB-100/300の使っているBASIC言語は、ほとんど共通していますが、本機の方が多くの命令を持っていますし、一部異なる点もあります。

●追加命令

PASS（プログラム保護）
BEEP（ブザー音）
READ（DATA文よりデータを読み込む）
DATA（データを書いておく）
RESTORE（読み込むデータの指定）
ON～GOTO（GOTO文の間接指定）
ON～GOSUB（GOSUB文の間接指定）
REM（注釈文）

●追加関数

DEG（60進数→10進数変換）
DMS$（10進数→60進数変換）
STR$（数値を数字に変換）

●変更命令

本機	PB-100/300
NEW(NEW ALL)	CLEAR(CLEAR ALL)
CLEAR	VAC
IF～THEN	IF～;
SAVE ALL	SAVE A
LOAD ALL	LOAD A
VERIFY	VER
DEFM(プログラム書き込み可)	DEFM(マニュアルのみ可)

●変更関数

本機	PB-100/300
KEY$ MID$	KEY MID

以上のような相異点がありますが、PB-100/300で作られたプログラムは、**原則としてそのまま使えます。**
実は、本機にはPB-100/300で作られたプログラムを判別する機能を持っています。ただし、使いやすくするためや、後からの見直しを便利にするために、本機用に手直しした方が良いと思います。

例)

PB-100用プログラム

```
10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80;B=B+1:GOTO 90
50 IF Z(A)<60;C=C+1:GOTO 90
60 IF Z(A)>40;D=D+1:GOTO 90
70 IF Z(A)>20;E=E+1:GOTO 90
80 F=F+1
90 NEXT A
   :
```

この例はデータを入力して、データの大きさに応じて振り分ける部分です。このようなプログラムでも、そのまま使えますが、本機用のプログラムにするには、次の点を直してください。
行番号10の"VAC"は"CLEAR"にする

```
10 CLEAR
```

行番号40から70の";"は"THEN"にする。

```
40 IF Z(A)>80 THEN B=B+1:GOTO 90
```
(以下同じ)

このプログラムでは変数の増設が必要ですので、PB-100/300ではマニュアルで実行していたDEFM命令を、プログラムの先頭に書き込みます。

```
5 DEFM 20
```

例2）

PB-100用プログラム

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
      :
      :
```

このプログラムは作業に合わせて分岐先を振り分けるプログラムですが、これを本機に合わせるには、ON～GOTO文を用いて、次のように変更します。

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
      :
      :
```

このようにON～GOTO文を使うことによりスッキリできますし、判断によりデータNをチェックする必要がなくなります。
以上の内容に注意すれば、今迄作られているPB-100/300用のプログラムがより有効に使えます。
すぐにでも色々なプログラムを使ってみたい方は、市販されているPB-100/300用のプログラム集を利用できますので、とても便利です。

なお、PBシリーズで作成され、テープに記録されているプログラムやデータは、そのまま本機で読み込むことができますが、本機で作成され、テープに記録されているプログラムやデータは、他のPBシリーズで読み込めないものもあります。これ等の関係を次に示しますので、注意してお使いください。

本機→PB-400, FX-710P

SAVE / LOAD	PF	AF	MF	パスワードつき		
				PF	AF	MF
LOAD	○			○		
LOAD ALL		○			○	

本機→PB-100, PB-200, PB-300, FX-700P, FX-802P

LOAD ＼ SAVE	PF	AF	MF	パスワードつき PF	パスワードつき AF	パスワードつき MF
LOAD	○		／	／	／	／
LOAD ALL		○	／	／	／	／

○：読み込めます。
（空欄）：通過ファイル名を表示しますが読み込めません。
／：通過ファイル名も表示せず、読み込めません。

〔注意〕

●本機で作成したプログラムを他のPBシリーズに移す場合は、プログラム中にREAD#、WRITE#、RESTORE#の命令があってはいけません。
また、KEY$、MID$はPB-100/200/300、FX-700P/802PではKEY、MIDとしてください。

●PB-100/300で作成されたプログラムを本機で実行した場合に、そのままでは正しく実行しないことがあります。
IF～THEN分岐先で、分岐先に数式が用いられている場合→エラー
⇨IF～THEN GOTO分岐先に訂正

第4章
コマンド・リファレンス

※ここからは、文法上の繰り返し等を説明するために、以下の記法を用います。

- **太字の語**——必ず、その通り書かなければいけない語。
- {××××／○○○○}—{ }の中の一つを選択して書かなければなりません。
- [○○○○]—[ ]の中は省略する書き方もあります。
- ○○○○*——右肩に＊がついた要素は繰り返して書くことができます。
- 数式——10、2＋3、A、S＊Q等の数値、計算式、数値変数。
- 文字式——〝ABC〟、X＄、N＄＋M＄等の文字定数、文字変数、文字計算式。
- パラメータ——コマンドに伴う要素。
- Ⓟ——プログラム中でのみ実行可能。
- Ⓜ——マニュアルでのみ実行可能。
- Ⓐ——プログラム中でもマニュアルでも実行可能。
- Ⓕ——関数命令。プログラム中でもマニュアルでも実行可能。

〈例〉

DATA〔データ〕〔**,**〔データ〕〕*

全ての要素に〔 〕がついていますから、“DATA”とだけ書くこともできます。また、,〔データ〕に〔 〕*がついていますから、この要素は繰り返して書いてもよいことになります。従って“DATA データ, データ, ……”と書くことができます。最初の〔データ〕の部分を省略すれば、“DATA ,データ,データ, ……”と書くこともできます。

GOTO {行番号／#プログラムエリア番号}

これは以下のような2通りの書き方を表わしています。

①GOTO 行番号
②GOTO #プログラムエリア番号

NEW 〔ALL〕 Ⓜ
(ニュー〔オール〕)

機能 プログラムの消去。プログラムと変数の消去。

パラメータ ALL指定したときはP0～P9の全プログラムと変数を消去します。

説明
(1)ALL指定がないときは、現在指定されているプログラムエリアのプログラムを消去します。変数は消去されません。

(2)ALL指定があるときは全プログラムエリアのプログラムと変数を消去します。DEFMによる設定も解除され、初期の26メモリーになります。

(3)パスワード設定中は実行できません。

(4)プログラム中に書き込んで使うことはできません。

(5)WRTモードでのみ実行できます。

※NEW ALLはNEW Aと省略することもできます。

例 MODE 1 NEW EXE

RUN 〔実行開始行〕 Ⓜ
行番号 (ラン)

機能 プログラムの実行。

パラメータ 実行開始行：行番号

説明
(1)指定した実行開始行(省略した場合はプログラムの先頭)からプログラムを実行します。

(2)指定した行番号が存在しないときは、指定より大きく、かつ一番近い行から実行を開始します。

(3)変数はクリアーされません。

例

```
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 END
```

```
RUN EXE          LINE 10
RUN 20 EXE       LINE 20
```

LIST〔{行番号 / ALL}〕 (リスト) Ⓜ

機能 プログラムの内容を表示します。

パラメータ 行番号：表示する最初の行番号。

ＡＬＬ：P0からP9までの全プログラム内容を順に表示します。

説明 Ｉ．RUNモードの場合

(1)行番号が指定されたときは指定行番号から、行番号が省略されたときは先頭の行から順にプログラム内容を表示します。

(2)プログラム内容は順に自動的に表示されますので、止めたいときは[STOP]キーを押します。再び次の行以降を表示させたいときは[EXE]キーを押します。

(3)プリンタ接続時のプリントモード("PRT"点灯中)では表示は止まらず順次、早い速度で表示します。

II．WRTモードの場合

(1)行番号が指定されたときは指定行番号から、行番号が省略されたときは先頭の行からプログラム内容を表示します。

(2)WRTモードでは1行ごとに表示して、編集可能状態となりますので、編集が必要ないときはそのまま[EXE]キーを押しますと次の行へ進みます。なお[SHIFT]キーに続けて押しますと、直前の行に戻ります。

●ALL指定があるときはP0から順にP9までの全プログラム内容を表示して行きます。このときはWRTモードでも順に送られ、編集することはできません。

●パスワード設定中は実行できません。

※LIST ALLはLIST Aと省略することもできます。

例

```
LIST [EXE]
LIST 30 [EXE]
```

PASS "パスワード" 文字列　(パス) Ⓜ

機能　パスワードを設定または解除します。

パラメータ　パスワード：１～８文字の文字列

説明

(1)パスワードが設定されていないときにこの命令を実行しますと、全プログラムエリア(P0～P9)にパスワードが設定されます。

(2)パスワードが設定されているときにこの命令を実行しますと、現在設定されているパスワードと後から実行したパスワードが一致したときのみ、パスワードが解除されます。パスワードが一致しないときはプロテクトエラー(ERR8)となります。

(3)パスワードは１～８文字の文字列で構成され、スペース、アルファベット、数字、特殊記号などが使えます。但し、「"」自身は使えません。

(4)パスワードが設定されているとき、LIST、LIST ALL、LIST#、NEW、NEW ALL、NEW#などのコマンドは使えず、またWRTモードでの行番号EXEもエラー(ERR8)となり実行できません。

(5)プログラム中では使えません。

(6)電源スイッチオフでもパスワードは保持されます。

(7)パスワードが設定されているときに、SAVEまたはSAVE ALL文によりプログラムをテープに記録しますと、パスワードも同時に記録されます。パスワードが設定されているプログラムを、テープからLOADまたはLOAD ALL文により読み込んだ場合は、プログラムについているパスワードが設定されます。なお、現在パスワードが設定されているときに、異なるパスワードのついたプログラムをテープから読み込むことはできません。(ERR8)

注意　パスワード設定後にパスワードを忘れてしまったときは、本体裏面のオールリセットボタンを押して、全プログラムとメモリーをクリアーしてください。

例　PASS "CASIO" EXE

SAVE 〔ALL〕 〔"ファイル名"〕 (セーブ〔オール〕) Ⓜ

文字列

機　能　プログラムをカセットテープに記録します。

パラメータ　ALL：全プログラムエリアのプログラムを記録。
ファイル名：1～8文字の文字列。省略可。

説　明
(1)ALLが省略された場合は、現在指定されているプログラムエリアの内容を記録します。

(2)ALLがついた場合は、P0からP9までの全プログラムエリアの内容を記録します。

(3)パスワードが設定されている場合は、パスワードをつけて記録しますので、LOADコマンドにより読み込んだときに同じパスワードがつきます。

※SAVE ALLはSAVE Aと省略できます。

例
SAVE EXE
SAVE "CASIO" EXE
SAVE ALL "PB" EXE

LOAD 〔ALL〕〔"ファイル名"〕 Ⓜ

文字列 (ロード〔オール〕)

機能 プログラムをカセットテープから読み込みます。

パラメータ ALL：全プログラムエリアのプログラムを読み込む。
ファイル名：1～8文字の文字列。省略可。

説明 (1)ALLが省略された場合は、現在指定されているプログラムエリアに"SAVE"で記録されたプログラムを読み込みます。

(2)ALLがついた場合は、P0からP9までのプログラムエリアに"SAVE ALL"で記録されたプログラムを読み込みます。

(3)パスワードつきで記録されたプログラムを読み込みますと、記録したときと同じパスワードが設定されます。

※LOAD ALLはLOAD Aと省略できます。

SAVEとLOADの関係

	LOAD	LOAD "ファイル名"	LOAD ALL	LOAD ALL "ファイル名"
SAVE	○	×	×	×
SAVE "ファイル名"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "ファイル名"	×	×	○	○

但し、ファイル名は同一のものです。 ○…読み込める
×…読み込めない

VERIFY〔"ファイル名"〕 Ⓜ
文字列

(ベリファイ)

機能 カセットテープ上のプログラムやデータの記録状態をチェックします。

パラメータ ファイル名：1～8文字の文字列。省略可。

説明 (1)ファイル名が指定された場合は、同一ファイル名のファイルをチェックします。

(2)ファイル名が省略された場合は、コマンド実行後に最初に現われたファイルをチェックします。

(3)チェック方式は、記録状態の型式をチェックします。(パリティーチェックといいます)

例 VERIFY EXE
VERIFY "PROG1" EXE

CLEAR Ⓐ

(クリアー)

機能 全ての変数をクリアーします。

説明 (1)全ての変数をクリアーし、数値変数には0を、文字変数にはヌル(何もない)を入れます。

(2)このコマンドはプログラム中に書き込んでも、マニュアルでも使えます。

(3)FOR・NEXTループ(120ページ参照)中ではループ制御変数もクリアーするために、NEXT文実行時にエラーとなります。

※CLEARコマンドはVACとしても同じに使えます。

END　（エンド）　Ⓟ

機　能　プログラムの実行を終了します。

説　明　プログラムの実行を終了しますので、次にプログラムがあっても実行しません。

STOP　（ストップ）　Ⓟ

機　能　プログラムの実行を一時停止します。

説　明　(1)プログラムの実行を一時的に停止し、"STOP"を表示して入力待ちとなります。

(2)停止はEXEキーにより実行を再開します。

(3)STOP文により停止しているときに、STOPキーを押しますと、プログラムエリアと行番号を表示します。

(4)STOPによる停止中はEXEキーによる計算が行なえます。

〔LET〕 {数値変数=数　式 / 文字変数=文字式} Ⓟ

(レット)

機　能　左辺の変数に右辺の式の値を代入します。

説　明　(1)数値型変数には数値式が、文字型変数には文字式が対応します。

(2)LETは省略することができます。

例

```
10 LET X=12
20 LET Y=X↑2+2*X-1
30 PRINT Y
40 A$="CASIO"
50 B$="PB&FX"
60 PRINT A$;B$
70 END
```

REM 注　釈 / 文字列 Ⓟ

(リマーク)

機　能　注釈を表わす文です。

説　明　(1)プログラム中に書き込み、REM以降は注釈文として扱われ何も実行されません。

(2)同一行内に実行させたいコマンドを書き込む場合は、REM文の前にマルチステートメント(:コロン)を書いてください。

例

```
10 INPUT "R",R
20 S=π*R↑2:REM MENSEKI
30 PRINT S
40 END
```

INPUT [″メッセージ文″,]変数名[,[″メッセージ文″,]変数名]* Ⓟ

(メッセージ文: 文字列)

(インプット)

機能 キーボードからデータを変数に入力します。

パラメータ メッセージ：文字列。
変数名: 数値変数名または文字変数名。

説明
(1)キーボードから指定した変数に入力します。

(2)メッセージがある場合、メッセージを表示し、続けて″?″を表示します。

(3)メッセージが省略された場合、″?″のみが表示されます。

(4)データ入力の最後にはEXEキーを押します。

(5)数値変数に文字データを入力しますとエラー(ERR2)となり、ACを押した後に再度″?″を表示してデータ入力を促します。但し、アルファベット1文字や数式を入力した場合、数式の結果が数値のときはその値が代入されます。

(6)データ入力待ちのときにEXEキーだけを押すと、ヌル（何もない)入力となり、変数が数値変数のときはエラー(ERR2)となります。

例

```
10 INPUT A
20 INPUT "B$=",B$
30 INPUT "C$=",C$,"D$=",D$
```

KEY$ (キーダラー) Ⓟ

機能 キーボードから1文字を入力する関数です。

説明 (1)キーボードからのキー入力を1文字分だけ受け入れます。

(2)数字、アルファベット、記号がキー入力できます。

(3)読み込まれたデータは1文字の文字型となります。

(4)"?"は表示せず、入力待ちにもなりませんので、通常はIF文と組み合わせて使います。

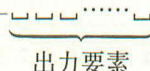
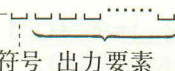
※KEY$はKEYと省略することもできます。

例

```
10 PRINT "INPUT<6>";
20 A$=""
30 K$=KEY$
40 IF K$="" THEN 30
50 A$=A$+K$
60 IF LEN(A$)<6 THEN 30
70 PRINT A$
80 END
```

● 6文字をキーボードから受けつけます。

PRINT 〔出力要素〕[{,;}〔出力要素〕]* Ⓟ (プリント)

機能 出力要素を表示します。

パラメータ 出力要素：出力制御関数(CSR)、数式、文字式。

説明 (1)出力要素を表示します。出力制御関数がつく場合は、それによって決められた位置から表示します。

(2)数式、文字式ではその値を表示します。

(3)出力要素が数式の場合は、値の前に符号桁(+, −)がつきますが、+符号は空白として表示されます。

●文字型表示——␣␣␣……␣ (出力要素)

●数値型表示——␣␣␣␣……␣ (符号 出力要素)

(4)出力要素が数式で仮数部が10桁以上の場合は、11桁目を四捨五入して表示します。また、仮数部以外に符号桁と指数部があるときは指数記号(E)と指数部2桁を表示します。

(5)出力要素と出力要素の区切りは , と ; が使え、 , のときは前の出力要素を表示後停止し(STOP点灯)、EXEキーにより、一度表示をクリアーしてから、次の出力要素を表示します。区切りが ; のときは前の出力要素に続けて次の出力要素を表示します。

(6)出力要素が全て省略されたとき(PRINTのみ)は、表示をクリアーするだけで停止はしません。

(7)MODE 7と押すプリントモードで印字中は、PRINT文を実行しても表示は停止しません。

(8)SET文により数値をフォーマット化することができます。

例

```
10 PRINT 1/3
20 PRINT "A="; A
30 PRINT "SIN 30", SIN 30
40 PRINT "END";
50 PRINT
60 END
```

CSR 出力位置指定 数式 (シーエスアール) Ⓕ

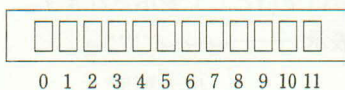
機　能　指定した位置から出力要素を表示します。

パラメータ　出力位置指定：数式で、値は小数点以下切り捨てとなります。

0≦指定＜12

説　明　(1)PRINT文中で用い、出力要素の出力位置を指定します。

(2)出力位置の与え方は左端を0とします。

0 1 2 3 4 5 6 7 8 9 10 11

例

```
10 FOR I=0 TO 11
20 PRINT CSRI;"A";CSR 11-I;"B"
30 NEXT I
40 END
```

● AとBの文字がEXEキーを押すごとに左右から移動します。

GOTO {分岐先行番号 / 行番号 | #プログラムエリア番号 / 0～9の1文字} Ⓟ

（ゴートゥー）

機　　能　指定された分岐先へ無条件で分岐します。

パラメータ　分岐先行番号：1～9999の行番号(1≦行番号<10000)
プログラムエリア番号：0～9の1文字

説　　明　(1)指定された分岐先へ分岐します。

(2)分岐先が行番号の場合は、現在のプログラムエリア内の指定行へ分岐し、プログラムを実行します。分岐先行番号が存在しない場合は、エラー(ERR 4)となります。

(3)分岐先がプログラムエリア番号の場合は、指定されたプログラムエリアへ分岐し、先頭からプログラムを実行します。

※分岐先行番号、プログラムエリア番号に数式も使えます。

例

```
 10 PRINT "START";
 20 GOTO 100
 30 PRINT "LINE 30"
 40 END
100 PRINT "LINE 100"
110 GOTO 30
```

ON 分岐条件 GOTO 〔分岐先〕〔,〔分岐先〕〕* Ⓟ

数 式

分岐先は { 分岐先行番号 / #プログラムエリア番号

(オン〜ゴートゥー)

機　　能　分岐条件に従って指定された分岐先へ分岐します。

パラメータ　分岐条件：数式で、値は小数以下切り捨てとなります。

分岐先行番号：1〜9999の行番号(1≦行番号<10000)

プログラムエリア番号：0〜9の1文字

説　　明　(1)分岐条件の式の値の整数部により分岐します。分岐先は先頭から順に式の値が1の場合、2の場合……と割り当てられます。

ON A GOTO 100, 200, 300, ………
(100: A＝1, 200: A＝2, 300: A＝3)

(2)式の値が1より小さいか、または相当する分岐先が書いてないときは分岐せずに、次の文を実行します。

(3)分岐先は一行に納まるまで、いくつでも書けます。

例

```
 10 INPUT A
 20 ON A GOTO 100,200,300
 30 PRINT "OTHER"
 40 GOTO 10
100 PRINT "LINE 100":GOTO 10
200 PRINT "LINE 200":GOTO 10
300 PRINT "LINE 300":GOTO 10
```

● 1〜3を入力すると100〜300行に分岐し、それ以外では"OTHER"を表示します。

IF 分岐条件（比較式） THEN {文〔：文〕* / 分岐先} Ⓟ

★分岐先は {分岐先行番号 / #プログラムエリア番号}

（イフ～ゼン）

機　能

分岐条件が成立したとき、THEN以降の文を実行します。またTHEN以降が分岐先の場合は分岐します。

パラメータ

分岐条件：比較式
分岐先行番号：1～9999の行番号(1≦行番号＜10000)
プログラムエリア番号：0～9の1文字

説　明

(1)分岐条件が成立したとき、THEN以降の文を実行または分岐先へ分岐します。

(2)分岐条件が成立しなかったときは、次の行を実行します。

(3)分岐条件は比較式(＝、≠、＜、＞、≦、≧)により判断します。

＝ 左辺と右辺が等しい	≠ 左辺と右辺が等しくない
＜ 左辺より右辺が大きい	＞ 左辺より右辺が小さい
≦ 左辺より右辺が大きいか等しい	≧ 左辺より右辺が小さいか等しい

(4) 2つ以上分岐条件がある場合は、THENの後にIF文を続けることができます。

IF～THEN IF～THEN………

※THENの後が文の場合は、THENのかわりに；が使えます。

例

```
10 N=6
20 PRINT CSR N;"↑";
30 K$=KEY$
40 IF K$="4"THEN N=N-1:IF N<0THEN N=0
50 IF K$="6"THEN N=N+1:IF N>11THEN N=11
60 PRINT
70 GOTO 20
```

●"↑"が④キーを押すと左に、⑥キーを押すと右に動きます。

FOR 制御変数名＝初期値(数式) TO 終値(数式)〔STEP 刻み幅(数式)〕Ⓟ NEXT 制御変数名（フォー～トゥー～ステップ・ネクスト）

機能　FOR文からNEXT文までの間を制御変数を初期値から終値まで刻み幅で変化させながら繰り返します。

パラメータ　制御変数名：単純変数名で、配列変数は使えません。
初期値：数式
終値：数式。
刻み幅：数式。省略したときは1の値

説明
(1)FOR文からNEXT文までの間を制御変数を初期値から終値まで、刻み幅で変化させながら繰り返し、制御変数が終値を超えたとき繰り返しを終了します。
(2)初期値が終値を超えている場合は、FOR～NEXTの間を1度だけ実行します。
(3)刻み幅は負数も使え、省略した場合は1となります。
(4)FOR文とNEXT文は必ず1対1で対応していなければなりません。また、FOR文に対応するNEXT文は、FOR文より後に書きます。
(5)FOR～NEXTのループは次のように入れ子構造にすることができます。

```
10 FOR I=1 TO 10
20 FOR J=11 TO 20
30 PRINT I;":";J
40 NEXT J
50 NEXT I
60 END
```

(6)入れ子構造にすることをネスティングともいい、4重までできます。
(7)FOR～NEXTループを終了したとき、制御変数は終値を超えたときの値となります。
(8)FOR～NEXTループからの外への飛び出しは可能ですが、IF文、GOTO文などでループ内へ飛び込むとエラーになります。なお、ループから飛び出したときも、ループの中であることを記憶していますので、NEXT文で終了させない限りネスティングを重ねていきます。

GOSUB { 分岐先行番号 / 行番号 / #プログラムエリア番号 / 0～9の1文字 } Ⓟ

(ゴーサブ)

機能 指定された分岐先へサブルーチン分岐します。

パラメータ 分岐先行番号：1～9999の行番号(1≦行番号<10000)

プログラムエリア番号：0～9の1文字

説明 (1)指定された分岐先へサブルーチン分岐します。サブルーチンからの戻りはRETURNの実行により行なわれます。

(2)サブルーチンの中からさらにサブルーチンを引用することをネスティングといい、8段までできます。

(3)RETURNによりGOSUB文の次の文に戻ります。

(4)GOSUB文によりサブルーチン分岐したときに、IF文やGOTO文などで次の文に戻すと、ネスティングは記憶されたままですので、必ずRETURNで戻してください。

(5)分岐先行番号が存在しない場合はエラー(ERR 4)となります。

※分岐先行番号、プログラムエリア番号に数式も使えます。

例

```
 10 PRINT "MAIN 10"
 20 GOSUB 100
 30 PRINT "MAIN 30"
 40 END
100 PRINT "SUB 100"
110 GOSUB 200
120 RETURN
200 PRINT "SUB 200"
210 RETURN
```

RETURN Ⓟ

(リターン)

機能 サブルーチンから復帰します。

説明 サブルーチンを呼んだ直後の文へ復帰します。

ON 分岐条件/数式 GOSUB〔分岐先〕〔，〔分岐先〕〕* Ⓟ

★分岐先は { 分岐先行番号 / #プログラムエリア番号

(オン〜ゴーサブ)

機　　能　分岐条件に従って指定された分岐先のサブルーチンへ分岐します。

パラメータ　分岐条件：数式で、値は小数以下切り捨てとなります。
分岐先行番号：1〜9999の行番号(1≦行番号<10000)
プログラムエリア番号：0〜9の１文字

説　　明　(1)分岐条件の式の値の整数部によりサブルーチン分岐します。
分岐先は先頭から順に式の値が１の場合、２の場合……と割り当てられます。

ON B GOSUB 1000, 2000, 3000……
(1000: B=1, 2000: B=2, 3000: B=3)

(3)式の値が１より小さいか、または相当する分岐先が書いてないときは分岐せずに、次の文を実行します。

(3)分岐先は一行に納まるまで、いくつでも書けます。

例

```
 10 INPUT A
 20 ON A GOSUB 100,200,300
 30 GOTO 10
100 PRINT "SUB 100":RETURN
200 PRINT "SUB 200":RETURN
300 PRINT "SUB 300":RETURN
```

●1〜3を入力すると各々のサブルーチンへ分岐します。

DATA 〔データ〕〔,〔データ〕〕*

定数　定数　Ⓟ　(データ)

機　能　データを収納します。

パラメータ　データ：文字定数または数値定数

説　明　(1)READ文で読み取るデータを書く為に用います。

(2)データは，で区切って複数個書くことができます。

(3)データ文だけを実行しても何もしません。

(4)文字定数の中に，を含むときは、データの両端を " で囲んでください。

DATA ABC, DEF, "GHI, JKL", ……
(ABC：1つ目、DEF：2つ目、"GHI, JKL"：3つ目)

(5)データを省略すると長さ0の文字列を表わします。

```
DATA A, ,B → DATA A,"",B
DATA ,     → DATA "",""
DATA       → DATA ""
```

READ 変数名〔,〔変数名〕〕* (リード) Ⓟ

機　能　DATAの内容を読み取ります。

パラメータ　変数名：数値変数または文字変数。配列変数も可。

説　明

(1)現在指定されているDATA文の中のデータを順に割り当て、指定された変数に代入します。

(2)数値変数には数値型のデータのみ読み取れます。

(3)DATA文の中のデータは、行番号の小さい方から大きい方へ、また同一DATA文中では先頭から順に読まれます。

(4)READ文で必要なデータを読んだ後、次のREAD文で読まれるのは、そのさらに後にあるデータです。

(5)プログラムの動作の初めには、どのデータも指定されません。READ文の最初の実行で、そのREAD文のあるプログラムエリアの先頭のデータが読まれ、以後はこのときのプログラムエリアのデータが順に読まれていきます。

(6)RESTORE文により読み込むデータの指定を変えることができます。

(7)READ文の変数よりDATA文中のデータが少ないときはエラー(ERR4)となります。

(8)DATA文中のデータの先頭にスペースがあるときは読みとばします。

例

```
10 DATA 1,2,3
20 READ A,B
30 PRINT A;B
40 DATA 4,5
50 READ C,D,E
60 PRINT C;D;E
70 END
```

●DATA文から順にデータを読み込み、表示します。

RESTORE 〔行番号〕 Ⓟ
数式
(レストア)

機能 READ文で読むデータの位置を指定します。

パラメータ 行番号：数式で、値は小数以下を切り捨てとなります。

1≦行番号＜10000

説明 (1)READ文で読むデータのあるDATA文を指定します。

(2)行番号を省略すると、データの指定を解除します。この後最初に実行するREAD文により、そのREAD文のあるプログラムエリアの先頭にあるデータが指定され、読まれます。

(3)行番号を指定すると、RESTORE文が存在するプログラムエリアの行番号が指定されます。以後のREAD文では、そのときのプログラムエリアのデータが次々に読まれます。

(4)指定された行番号が存在しないときや指定された行番号以降にDATA文が存在しないときは、エラー(ERR4)となります。

例

```
10 DATA 1,2,3
20 DATA 4,5
30 READ A,B,C,D,E
40 RESTORE 10
50 READ F,G
60 RESTORE 20
70 READ H,I
80 PRINT A;B;C;D;E;F;G;H;I
90 END
```

PUT 〔"ファイル名"〕変数1〔，変数2〕 Ⓐ

文字列　（プット）

機能 カセットテープにデータを記録します。

パラメータ ファイル名：1～8文字の文字列。省略可。
変数1，変数2：記録する変数範囲の指定

説明 (1)カセットテープに変数の内容を記録します。
(2)変数の指定は次のように書きます。

PUT　A……………………変数Aの内容
PUT　A,Z…………………変数A～Zの内容
PUT　A,A(100)…………変数A～A(100)の内容
PUT　$,D,W……………専用文字変数$とD～Wの内容

専用文字変数$の内容を記録する場合は$を最初に書き込みます。
(3)マニュアルでもプログラム中に書き込んでも使えます。

GET 〔"ファイル名"〕変数1〔，変数2〕 Ⓐ

文字列　（ゲット）

機能 カセットテープに記録されているデータを変数に読み込みます。

パラメータ ファイル名：1～8文字の文字列。省略可。
変数1，変数2：読み込む変数の指定

説明 (1)カセットテープに記録されているデータを指定された変数に読み込みます。
(2)変数の指定は次のように書きます。

GET　A……………………変数Aに読み込む
GET　A,Z…………………変数A～Zに読み込む
GET　A,A(100)…………変数A～A(100)に読み込む
GET　$,D,W……………専用文字変数$とD～Wに読み込む

(3)PUTにより記録した変数名とGETにより読み込む変数名は一致していなくてもかまいません。
(4)読み込もうとする変数より記録されているデータが少ない場合は、記録されているデータだけ、先頭の変数から順に読み込みます。
(5)マニュアルでもプログラム中に書き込んでも使えます。

BEEP [{0|1}] (ビープ) Ⓐ

機能　ビープ音を発生させます。

パラメータ　0：低音。
1：高音。
省略した場合は0と同じ。

説明　(1)低い音と高い音のビープ音を発生させます。
(2)マニュアルでも使えます。

例

```
 10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ":N=0
 20 FOR I=1 TO 10
 30 A$=MID$( RAN#*26+1,1 )
 40 PRINT CSR4;"<";A$;">";
 50 FOR J=1 TO 30
 60 K$=KEY$:IF K$≠""THEN80
 70 NEXT J
 80 IF K$=A$ THEN BEEP 1:N=N+1:GOTO 100
 90 BEEP 0
100 PRINT:NEXT I
110 PRINT N;
120 IF N>10 THEN END
130 FOR I=1 TO 10
140 BEEP 0:BEEP 1
150 NEXT I
```

●表示された文字に対応するアルファベットキーを押します。

DEFM 〔増設メモリー数〕 Ⓐ
数式　(ディーイーエフエム)

機能　メモリーの増設をします。

パラメータ　増設メモリー数：数式で、値は小数以下切り捨てとなります。
省略可。0≦増設メモリー数<453。(RC-4装着時)

説明
(1)メモリー(変数エリア)の増設を行ないます。

(2)増設メモリー数は1個単位で残りステップ数に応じて任意に指定できます。

(3)メモリー1つ増設するごとに8ステップ必要となりますので、プログラムエリアのステップ数は減少します。

(4)配列変数を使用し、データメモリーを多く必要とするときに使用します。

(5)増設メモリー数を省略した場合は、現在設定されているメモリー数を表示します。

(6)マニュアルでもプログラム中に書き込んでも使え、マニュアルで実行した場合は新しい設定状態(増設メモリー数+基本メモリー数26)を表示します。プログラム中に書き込んで実行した場合は新しい設定状態は表示されません。

(7)残りステップ数よりも多くのメモリーを増設しようとした場合は、エラー(ERR1)となります。

(8)増設したメモリーをクリアーにして、最初の26個に戻すには、DEFM 0を指定します。

例

DEFM 10 EXE　→　`***VAR:36`

DEFM EXE　→　`***VAR:36`

```
10 DEFM 10
20 FOR I=1 TO 10
30 INPUT Z(I)
40 NEXT I
  :
```

MODE 数式 (モード) Ⓟ

機　能　計算機の状態を設定します。

パラメータ　数式：値は小数以下切り捨てとなります。

4 ≦数式＜ 9

説　明

(1)数式の値により角度単位やプリントモードの設定・解除を設定します。

(2)設定は次の通りです。

MODE4………角度単位を度に設定します。
MODE5………角度単位をラジアンに設定します。
MODE6………角度単位をグレードに設定します。
MODE7………〝PRT〟を表示し、プリントモードに設定します。
MODE8………プリントモードを解除します。

(3)これは[MODE]キーによる設定と同じですが、RUNモードやWRTモードの設定はできません。

(4)入力方法は[MODE]キーではなく、[M][O][D][E]とアルファベットキーを使います。

例

```
10 MODE 4
20 A=SIN 30
30 MODE 7
40 PRINT A
50 MODE 8
60 END
```

SET {Fn / En / N}

★nは0～9の整数　　(セット)　Ⓐ

機能　数値データの出力形式(フオーマット)を指定します。

パラメータ　Fn：小数点以下指定。
En：有効桁数指定。
N　：指定解除。

説明　(1)数値データを出力するときの、小数点以下の桁数や有効桁数の指定を行ないます。

(2)小数点以下指定(Fn)では、小数点以下の桁数を0から9桁まで指定します。

(3)有効桁数指定(En)では、有効桁数を1から10桁まで指定します。なお、"**SET E0**"としたときは10桁指定となります。

(4)"**SET N**"では両指定を解除します。

(5)マニュアルでもプログラム中に書き込んでも使えます。

例

```
10 INPUT N
20 SET F5:PRINT N
30 SET E5:PRINT N
40 SET N:GOTO 10
```

LEN（単純文字変数）　(レングス)　Ⓕ

機　能　与えられた単純文字変数の中の文字数を値とする関数です。

パラメータ　単純文字変数：配列変数は使えません。

説　明　(1)単純文字変数の中の文字数を数える関数です。

(2)使える文字変数は単純文字変数(A$, Y$等)で、配列文字変数(B$(3)等)は使えません。

例

```
10 INPUT A$
20 PRINT LEN(A$)
30 GOTO 10
```

MID$ (位置〔, 文字数〕) Ⓕ

位置: 数式　文字数: 数式

(ミッドダラー)

機能

専用文字変数($)に記憶されている文字列の、指定した位置から指定文字数を取り出す関数です。

パラメータ

位置：数式で、値は小数点以下を切り捨てます。

1 ≦位置＜101

文字数：数式で、値は小数点以下を切り捨てます。

1 ≦文字数＜101

省略すると、位置以降の全てが指定されます。

説明

(1)専用文字変数($)に記憶されている文字列の、指定した位置から指定した文字数分の文字を取り出す関数です。

(2)位置が文字列の範囲を超えて指定されたとき（文字数より位置の方が大きいとき)は、ヌル(何もない)を与えます。

(3)文字列の、指定した位置以降の長さが指定した文字数より小さいときは、指定した位置以降の文字列全部を取り出します。

※MID $はMIDと省略することができます。

例

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 INPUT M,N
30 PRINT MID$(M,N)
40 END
```

VAL (単純文字変数) Ⓕ

(バリュー)

機能 単純文字変数内の数字を数値に変換する関数です。

パラメータ 単純文字変数：配列変数は使えません。

説明 (1)単純文字変数内の数字を数値に変換します。

(2)文字変数の内容が数字のみ(+、−、・、E、E⁻を含む)の場合は、そのまま数値に変換します。

A＄＝"−12.3"のとき、VAL(A＄)→−12.3

(3)文字変数の内容が数字以外で始まっている場合は、エラーとなります。

A＄＝"A45"のとき、VAL(A＄)→エラー(ERR 2)

(4)文字変数の内容が数字で始まっているが、途中に数字以外が入っている場合は、最初の数字部分を数値に変換します。

A＄＝"78A9"のとき、VAL(A＄)→78

例

```
10 INPUT A$
20 PRINT VAL(A$)
30 END
```

STR$ (数式) Ⓕ

(ストリングダラー)

機能 数式の値を文字(数字)に変換します。

パラメータ 数式：数値、計算式、数値変数、配列数値変数

説明 (1)数式の値を文字に変換します。

(2)数式が計算式の場合は、計算結果を文字に変換します。

(3)数式が正の場合は、符号桁は削除され、数字だけとなります。

例

```
10 PRINT STR$(123)
20 PRINT STR$(45+78)
30 A=963
40 PRINT STR$(A)
50 END
```

数値関数

SIN 引数/数式　COS 引数/数式　TAN 引数/数式　Ⓕ

機能　与えられた引数に対する三角関数の値を与える関数です。

パラメータ　引数：数式。

－1440°＜引数＜1440°　（度）

-8π＜引数＜8π　（ラジアン）

－1600＜引数＜1600（グレード）

但し、TANにおいては |引数|＝$(2n-1)$＊(1直角)を除く

1直角＝90°＝$\frac{\pi}{2}$rad＝100grad

説明　(1)与えられた引数に対する三角関数の値を与える関数です。

(2)値は角度単位の設定(MODEキー、モードコマンド)に従います。

ASN 引数/数式　ACS 引数/数式　ATN 引数/数式　Ⓕ

機能　与えられた引数に対する逆三角関数の値を与える関数です。

パラメータ　引数：数式。ASN、ACSは-1≦引数≦1。

説明　(1)与えられた引数に対する角度を与える逆三角関数です。

(2)値は角度単位の設定(MODEキー、モードコマンド)に従います。

(3)関数の値は以下の範囲で与えられます。

－90°≦ASN X ≦90°

0°≦ACS X ≦180°

－90°≦ATN X ≦90°

LOG 引数/数式　LN 引数/数式 Ⓕ

機　能　対数関数の値を与える関数です。

パラメータ　引数：数式。 0 <引数。

説　明　対数関数の値を与えます。

- ●LOG　常用対数関数　$\log_{10} x$、$\log x$
- ●LN　自然対数関数　$\log_e x$、$\ln x$

EXP 引　数/数式 Ⓕ

機　能　指数関数の値を与える関数です。

パラメータ　引数：数式。－227≦引数≦230。

説　明　指数関数の値を与える関数です。

EXP　e^x

SQR 引　数/数式 Ⓕ

機　能　引数の平方根を与える関数です。

パラメータ　引数：数式。 0≦引数。

説　明　引数の平方根を与える関数です。

SQR　$\sqrt{x}$

ABS 引　数　数式 Ⓕ

機能	引数の絶対値を与える関数です。
パラメータ	引数：数式。
説明	引数の絶対値を与える関数です。

ABS　　$|x|$

SGN 引　数　数式 Ⓕ

機能	引数の符号に応じた値を与える関数です。
パラメータ	引数：数式。
説明	引数の符号に応じた値を与える関数です。

引数が正の場合、1
引数が０の場合、０
引数が負の場合、-1

INT 引　数　数式 Ⓕ

機能	引数を越えない最大の整数を与える関数です。
パラメータ	引数：数式。
説明	引数を越えない最大の整数を与える関数です。

INT 12.56→12
INT -78.1→-79

FRAC 引数 Ⓕ

（引数：数式）

機　能	引数の小数部を値とする関数です。
パラメータ	引数：数式。
説　明	引数の小数部を値とする関数です。符号は引数の符号と一致します。

RND (引数 ,桁位置) Ⓕ

（引数：数式、桁位置：数式）

機　能　引数を指定した桁で四捨五入した値を与える関数です。

パラメータ　引数：数式。
桁位置：数式。値は小数点以下切り捨てとなります。

−100＜桁指定＜100

説　明　(1)引数を指定した桁で四捨五入した値を与える関数です。

(2)桁指定は10の桁位置乗の位置を四捨五入します。

小数点以下3桁目を四捨五入→RND(x, −3)
100の位を四捨五入――――→RND(x, 2)

RAN# Ⓕ

機　　能　0から1の間の乱数を与える関数です。

説　　明　(1) 0から1の間の乱数を与える関数です。0＜乱数＜1

(2)乱数は小数点以下10桁です。

例

0～9の1桁の乱数を作る

```
INT(RAN#*10)
```

1～5の1桁の乱数を作る

```
INT(RAN#*5)+1
```

10～99の2桁の乱数を作る

```
INT(RAN#*90)+10
```

DEG (度〔,分〔,秒〕〕) Ⓕ

度：数式　分：数式　秒：数式

機　　能　60進数を10進数に変換します。

パラメータ　度：数式。

分：数式。

秒：数式。

|DEG (度,分,秒)| ＜ 10^{100}

説　　明　度・分・秒で示される60進数を10進数に変換します。

例　DEG(12,34,56) EXE　　`12.5822222`

```
10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS$ (引　数) Ⓕ
数式

機　　能　10進数を60進数に変換します。

パラメータ　引数：数式。 |数式|＜10^{100}

説　　明　(1)10進数を60進数に変換します。

(2)変換された結果は、文字列として与えられます。

例　DMS$(45.678) EXE　　45°40'40.8

```
10 INPUT A
20 $=DMS$(A)
30 PRINT$
40 END
```

データバンク用コマンド

NEW＃ Ⓜ

（ニュークロスハッチ）

機　能　データバンクデータの消去

説　明　(1)データバンクに記憶されているデータを全て消します。

(2)パスワード設定中は実行できません。

(3)WRTモードでのみ実行できます。

例　MODE 1
NEW＃EXE
MODE 0

LIST＃ Ⓜ

（リストクロスハッチ）

機　能　データバンクデータを全て表示します。

説　明　(1)データバンクに記憶されているデータを記録された順に表示します。

(2)表示される内容はレコードナンバー(記録された順番)とメモデータです。

(3)データバンクデータは順に自動的に表示されますので、止めたいときはSTOPキーを押します。再び次を表示させたいときはEXEキーを押します。

(4)プリントモード(MODE 7と押す)では表示は止まらず、順次早い速度で表示します。

(5)パスワード設定中は実行できません。

(6)メモインモード(MODE 9と押す)では実行できません。

例　LIST＃EXE

SAVE# 〔"ファイル名"〕 Ⓜ
文字列　　（セーブクロスハッチ）

機　能　データバンクデータをカセットテープに記録します。

パラメータ　ファイル名：1～8文字の文字列。省略可。

説　明

(1)データバンクに記憶されているデータを全て、カセットテープに記録します。

(2)SAVE、SAVE ALLではデータバンクデータを記録しませんので、メモデータは必ずこのSAVE#で記録してください。

(3)パスワードが設定されている場合は、パスワードをつけて記録しますので、LOAD#コマンドにより読み込んだときに同じパスワードがつきます。

(4)メモインモードでは実行できません。

例

```
SAVE# EXE
SAVE#"CASIO" EXE
```

LOAD# 〔"ファイル名"〕 Ⓜ
文字列　　（ロードクロスハッチ）

機　能　データバンクデータをカセットテープから読み込みます。

パラメータ　ファイル名：1～8文字の文字列。省略可。

説　明

(1)カセットテープに記録されているデータバンクデータを読み込みます。

(2)パスワードつきで記録されたデータバンクデータを読み込みますと、記録したときと同じパスワードが設定されます。

(3)すでにデータが入っている場合は、前のデータをクリアーしてから新たなデータを読み込みます。

(4)メモインモードでは実行できません。

例

```
LOAD# EXE
LOAD#"CASIO" EXE
```

READ# 変数名〔,変数名〕* (リードクロスハッチ) Ⓟ

機能 データバンクデータを読み取ります。

パラメータ 変数名：数値変数または文字変数。配列変数も可。

説明
(1)データバンクに記憶されているデータを、順に変数に読み込みます。
(2)数値変数には数値型データのみ読み取られ、文字型データの場合はエラー(ERR2)となります。
(3)READ#で必要なデータを読んだ後、次のREAD#で読み込まれるのは、さらにその後にあるメモデータです。
(4)データバンクデータが,で区切られているときは、,ごとにデータを読みます。

例）データ

```
No.1  A,X,Y
No.2  B,Z
No.3  C
```

⬇

読み込む順番

A→X→Y→B→Z→C

(5)読むべきデータがない場合はエラー(ERR4)となります。
(6)RESTORE#(143ページ参照)により読み込むデータの順番を変更することができます。
(7)データバンクデータの先頭にスペースがある場合は、スペースを読み飛ばします。
(8)データバンクデータが" "(ダブルクォーテーション)で囲まれているときは、" "の中の文字列を読み込みます。

例

<メモデータ>

```
No.1  1,2,3
No.2  4,5,6
No.3  7,8,9
No.4  10,
```

<プログラム>

```
10 A=0
20 READ#$
30 IF $="" THEN 60
40 A=A+VAL($)
50 GOTO 20
60 PRINT "Σx=";A
70 END
```

●電子メモから数値データを読み込み、合計を求めます。

RESTORE# 〔"検索文字列"〔,〔{0/1}〕〔,{行番号/#プログラムエリア番号}〕〕〕 Ⓟ

文字式

(レストアクロスハッチ)

機能 データバンクデータを検索し、READ#で読むメモデータの順番を設定します。

パラメータ

検索文字列　　　　　：文字式。文字列の場合は" "で囲みます。

行番号　　　　　　　：数式。0＜行番号＜10000

プログラムエリア番号：数式。0≦プログラムエリア番号＜10

説明

(1)データバンクデータを検索し、次のREAD#で読むデータの順番を設定します。

(2)パラメータとデータ検索の関係は次のようになっています。

①RESTORE#

検索文字列以降が省略された場合には、次のREAD#で読むデータは先頭からとなります。

②RESTORE#"検索文字列"

検索文字列を先頭に含むデータを検索し、該当するデータが次のREAD#で読まれます。

③RESTORE#"検索文字列",{0/1}

0が指定された場合は②(省略)と同じになります。

1が指定された場合は、検索したデータを含む行の先頭データが、次のREAD#で読まれます。

④RESTORE#"検索文字列",〔{0/1}〕,{行番号/#プログラムエリア番号}

検索を実行して該当するメモデータがない場合、指定された行番号またはプログラムエリアに分岐します。

※②および③において、該当するメモデータがない場合はエラー(ERR4)となります。

※④で、分岐先行番号が存在しないときや、分岐先プログラムエリアにプログラムが存在しないときはエラー(ERR4)となります。

例

〈メモデータ〉

No.1 SUZUKI,347-4811,SHINJUKU
No.2 KOMATSU,045-211-0821,YOKOHAMA
No.3 SATO,06-314-2681,OSAKA
No.4 SHIMIZU,075-351-1161,KYOTO

〈プログラム〉

```
 10 RESTORE#              } 先頭に記憶されているデータを表示します。
 20 READ# $
 30 PRINT $
 40 RESTORE#"K"           } 頭文字がKであるデータを表示します。
 50 READ# $
 60 PRINT $
 70 RESTORE#"KY",1        } 先頭2文字がKYであるデータを検索し、そのデータのある行の先頭データを表示します。
 80 READ# $
 90 PRINT $
100 RESTORE#"AA",1,200    } 先頭2文字がAAであるデータがないときは行番号200へ分岐します。
110 READ# $
120 PRINT $
130 END
200 PRINT"MEMO END"
210 END
```

```
RUN EXE    SUZUKI
    EXE    KOMATSU
    EXE    SHIMIZU
    EXE    MEMO END
```

WRITE# 〔メモデータ式〔，メモデータ式〕*〕 Ⓟ

（ライトクロスハッチ）

機　　能　データバンクデータの書きかえまたは削除をします。

パラメータ　メモデータ：数式および文字式。文字列の場合は" "で囲みます。

説　　明

(1)現在RESTORE＃等で指定されているレコードナンバーのデータバンクエリアにメモデータを書き込みます。

(2)該当レコードのデータの有無にかかわらず、新たに書き込まれます。

(3)メモデータが全て省略されて実行されたときは、そのレコードにあったメモデータを消去します。

(4)メモデータが複数個あるときは、，で区切って書くことができます。このときのデータは，といっしょに一行に書き込まれます。

(5)WRITE＃で書き込まれた後のレコードナンバー指定は、書き込まれた次のメモデータとなります。

例

```
 10 REM WRITE
 20 RESTORE#             ┐
 30 WRITE#"A,B,C"        ┘──── 新たなデータを書きます。
 40 RESTORE#
 50 FOR I=1 TO 3
 60 READ# $:PRINT $;
 70 NEXT I
 80 PRINT" "
 90 REM CHANGE
100 RESTORE#             ┐
110 FOR I=1 TO 3         ├──── データを書きかえる。
120 WRITE# STR$(I)       ┘
130 NEXT I
140 RESTORE#
150 FOR I=1 TO 3
```

```
160 READ# $: PRINT $;
170 NEXT I
180 PRINT" "
190 REM CLEAR
200 RESTORE#
210 WRITE#                    ――――――データの消去
220 RESTORE#
230 READ# $
```

操 作	表 示
RUN EXE	ABC
EXE	123
EXE	ERR4 P0-230

↓
データ消去のためデータ不足

第5章
ライブラリー

統計計算

このプログラムは、１変数の標準偏差計算と、２変数の回帰分析の両方ができます。
使い方は簡単で、データを入力するだけで答えが求められます。データ数はいくつでもかまいません。計算式は次のようになっています。

n：データ数、Σx：xデータの総和、Σy：yデータの総和
Σx^2：xデータの２乗和、Σy^2：yデータの２乗和、Σxy：データの積和

xデータの平均($\bar{x}$)　：$\dfrac{\Sigma x}{n}$

yデータの平均($\bar{y}$)　：$\dfrac{\Sigma y}{n}$

xデータの標準偏差($x\sigma_{n-1}$)：$\sqrt{\dfrac{n\Sigma x^2-(\Sigma x)^2}{n(n_{-1})}}$　［集団中のサンプルデータを使う場合］

xデータの標準偏差($x\sigma_n$)　：$\sqrt{\dfrac{n\Sigma x^2-(\Sigma x)^2}{n^2}}$　［有限母集団全部のデータを使う場合］

yデータの標準偏差($y\sigma_{n-1}$)：$\sqrt{\dfrac{n\Sigma y^2-(\Sigma y)^2}{n(n_{-1})}}$

yデータの標準偏差($y\sigma_n$)　：$\sqrt{\dfrac{n\Sigma y^2-(\Sigma y)^2}{n^2}}$

一次回帰定数項(A)　：$\dfrac{\Sigma y-\mathrm{LRB}\cdot\Sigma x}{n}$

一次回帰係数(B)　：$\dfrac{n\cdot\Sigma xy-\Sigma x\cdot\Sigma y}{n\cdot\Sigma x^2-(\Sigma x)^2}$

相関係数(r)　：$\dfrac{n\cdot\Sigma xy-\Sigma x\cdot\Sigma y}{\sqrt{\{n\Sigma x^2-(\Sigma x)^2\}\{n\Sigma y^2-(\Sigma y)^2\}}}$

Xの推定値($\hat{x}$)　：$\dfrac{y_n-\mathrm{LRA}}{\mathrm{LRB}}$

yの推定値($\hat{y}$)　：$\mathrm{LRA}+x_n\cdot\mathrm{LRB}$

●プログラムリスト

```
 10 PRINT "START ?(
    Y/N)";
 20 $= KEY$
 30 IF $="N" THEN 1
    00
 40 IF $≠"Y" THEN 2
    0
 50 PRINT : BEEP 0
 60 CLEAR
 70 PRINT "DATA 1 0
    R 2?";
 80 A$= KEY$
 90 IF A$≠"1" THEN
    IF A$≠"2" THEN
    80
100 B=B+1
110 PRINT : BEEP 1
120 PRINT "X DATA";
    B;
130 INPUT X$: BEEP
    1
140 IF X$="E" THEN
    B=B-1: BEEP 0:
    GOTO 240
150 C=C+ VAL(X$)
160 D=D+ VAL(X$)↑2
170 IF A$="1" THEN
    100
180 PRINT "Y DATA";
    B;
190 INPUT Y
200 H=H+Y
210 I=I+Y*Y
220 M=M+ VAL(X$)*Y
230 GOTO 100
240 E=C/B
250 F= SQR((B*D-C*C
    )/(B*(B-1)))
260 G= SQR((B*D-C*C
    )/(B*B))
270 IF A$="1" THEN
    340
280 J=H/B
290 K= SQR((B*I-H*H
    )/(B*(B-1)))
300 L= SQR((B*I-H*H
    )/(B*B))
310 O=(B*M-C*H)/(B*
    D-C*C)
320 N=(H-O*C)/B
330 P=(B*M-C*H)/ SQ
    R((B*D-C*C)*(B*
    I-H*H))
340 INPUT "INPUT(0-
    17)",Z
350 IF Z=0 THEN PRI
    NT "END";: END
360 ON Z-15 GOTO 45
    0,480
370 RESTORE
380 FOR W=1 TO Z
390 READ V$
400 NEXT W
410 PRINT V$;"=";A(
    Z)
420 DATA N,SUMX,SUM
    X2,MEANX,SDX,SD
    XN,SUMY
430 DATA SUMY2,MEAN
    Y,SDY,SDYN,SUMX
    Y,LRA,LRB,COR
440 GOTO 340
450 INPUT "Y DATA",
    Y
460 PRINT "EOX=";(Y
    -N)/O
470 GOTO 340
480 INPUT "X DATA",
    X
490 PRINT "EOY=";N+
    X*O
500 GOTO 340
```

計 728steps

●メモリー

A		1変数と2変数の判断	L	A(11)	Yデータの標準偏差($y\sigma_n$)
B	A(1)	データ数	M	A(12)	データの積和
C	A(2)	Xデータの総和	N	A(13)	一次回帰定数項
D	A(3)	Xデータの2乗和	O	A(14)	一次回帰係数
E	A(4)	Xデータの平均	P	A(15)	相関係数
F	A(5)	Xデータの標準偏差($x\sigma_{n-1}$)	V$		出力名用
G	A(6)	Xデータの標準偏差($x\sigma_n$)	W		FORループ用
H	A(7)	Yデータの総和	X		Xデータ入力用
I	A(8)	Yデータの2乗和	Y		Yデータ入力用
J	A(9)	Yデータの平均	Z		出力選択用
K	A(10)	Yデータの標準偏差($y\sigma_{n-1}$)	$		KEY$関数用

では実際に使ってみましょう。
例として、次のデータを使ってみます。

	1	2	3	4	5
x(温度)	10	15	20	25	30
y(鋼棒)	1003	1005	1010	1011	1014

操　作	表　示
RUN EXE	START ?(Y/N)

新規データか、継続かを聞いてきますので、新規のときはYキーを押します。

Y	DATA 1 OR 2?

1変数統計か2変数統計かを聞いてきます。この例では2変数ですので2キーを押します。

2	X DATA 1?

ここからは順番にX・Yデータを入力します。

データの入力が終りましたら、終了サインとして「E」(ENDの意味)を入力します。

E EXE (# E の方です)　　INPUT(0-17)?

ここでは答えの表示を選択するコード番号(0～17)を入力します。コード番号はこの例題の後に記載しています。
まずは両データの平均を求めてみます。

	キー	表示
($\bar{x}$)	4 EXE	MEANX= 20
	EXE	INPUT(0-17)?
($\bar{y}$)	9 EXE	MEANY= 1008.6
	EXE	INPUT(0-17)?

次に定数項、係数、相関係数を求めてみます。

(A) 13 EXE
EXE
(B) 14 EXE
EXE
(r) 15 EXE
EXE

次に y が1000のときの推定値 $\hat{x}$ と x が18のときの推定値 $\hat{y}$ を求めます。

	キー	表示
	16 EXE	Y DATA?
(y_n)	1000 EXE	EOX= 4.642857143
	EXE	INPUT(0-17)?
	17 EXE	X DATA?
	18 EXE	EOY= 1007.48
	EXE	INPUT(0-17)?

計算を終了させるには0を入力します。

0 EXE

```
END
```

※表示部分で左側の点線内は、順に表示が送られて消えていきます。

この後に続けてデータを入力したいときは、プログラム実行後Nを押します。

RUN EXE
N

```
START ?(Y/N)
X DATA 6?
```

また、1変数統計のときは次のようになります。

RUN EXE
Y
1
10 EXE
15 EXE

```
START ?(Y/N)
DATA 1 OR 2?
X DATA 1?
X DATA 2?
X DATA 3?
```

コード番号表

コード		コード	
1	データ数(n)	10	yの標準偏差($y\sigma_{n-1}$)
2	Xの総和(Σx)	11	yの標準偏差($y\sigma_n$)
3	Xの2乗和(Σx^2)	12	データの積和(Σxy)
4	Xの平均($\bar{x}$)	13	一次回帰定数項(A)
5	xの標準偏差($x\sigma_{n-1}$)	14	一次回帰係数(B)
6	xの標準偏差($x\sigma_n$)	15	相関係数(r)
7	yの総和(Σy)	16	xの推定値($\hat{x}$)
8	yの2乗和(Σy^2)	17	yの推定値($\hat{y}$)
9	yの平均($\bar{y}$)		

〈ポイント〉

このプログラムは変数をB～Pまでの通常の使い方と、A(1)～A(15)までの配列変数との２通りに扱っています。
これは、Bという変数はA(1)という配列変数と同じ箱を使っていますので、呼び方を変えても同じ内容が入っているのです。つまり、行番号330までは計算式が異なりますので、BやC、D……というように１文字の変数として扱っています。しかし、行番号340からはコード番号を入力して、該当する答えを呼び出した方が、プログラムも短くスッキリとでき上りますので、配列変数の呼び名を使っています。
このような使い方は、応用といて便利ですが、変数の使い方をまちがえると大変なことになりますので、注意が必要です。

万能たてよこ集計

このプログラムは6個の独立したプログラムからできています。

P0に入れるプログラムは**「データ入力プログラム」**で、表のたてとよこを指定して、データを入力します。

P1に入れるプログラムは**「表示(印字)プログラム」**で、表の中のデータを順次表示または印字します。

P2に入れるプログラムは**「データ編集用プログラム」**で、一度記憶させたデータの訂正を行ないます。

P3に入れるプログラムは**「計算プログラム」**で、たて計とよこ計、そして総合計を求めます。

P4に入れるプログラムは**「テープ記録用プログラム」**で、メモリー内のデータをカセットテープに記録します。

P5に入れるプログラムは**「テープからの読み込み用プログラム」**で、カセットテープに記録してあるデータをメモリー内に読み込みます。

では、次のデータを使って操作してみましょう。

	1	2	3	4	5	6
1	376	159	248	767	311	351
2	320	85	287	833	291	541
3	480	41	166	750	426	367
4	518	269	343	565	221	268
5	536	158	426	495	235	492

操作	表示	
SHIFT P0	New[Y/N]?	…新たに表を作るときはYを押します。
Y	TATE?	…たての項目数を入力します。
5 EXE	YOKO?	…よこの項目数を入力します。
6 EXE	(1, 1)?	…順番にデータを入力します。
376 EXE	(1, 2)?	
159 EXE	(1, 3)?	
248 EXE	(1, 4)?	
⋮	⋮	
235 EXE	(5, 6)?	
492 EXE	END	…データ入力の終了です。

次に、入力したデータを確認してみましょう。

P2に入れた編集用のプログラムは、入力したデータがまちがっていたときや、一部のデータを変更して計算をしたいときに使います。
では、たてが3で、よこが4のデータを"450"とまちがえていたとします。

操　作	表　示	
SHIFT P2	TATE?	…たての位置を指定します。
3 EXE	YOKO?	…よこの位置を指定します。
4 EXE	(3, 4) 450?	…たて3、よこ4のデータを表示しますので訂正が必要なときは、正しい数値を入力します。
750 EXE	(3, 5) 462?	

以後、次のデータを見たいときは[+][EXE]と押し、前のデータを見たいときは[−][EXE]と押します。これで、前後のデータも訂正できます。

+ EXE	(3, 6) 367?
+ EXE	(4, 1) 518?

訂正が終りましたら[=][EXE]と押せば、最初の"TATE？"に戻り、たてとよこの指定になりますし、"TATE？"と表示されているときに[=][EXE]と押せば、このプログラムは終了します。

= EXE	TATE?
= EXE	END

なお、データが表示されているときに、数値を入力しますと新たな数値の入力となりますので、注意してください。

P3に入れたプログラムは、たて計とよこ計、そして総合計を求めます。

P4とP5に入れたプログラムはデータ保存用で、カセットインタフェイス〈FA-3〉をお持ちの方で、入力したデータを保存したいときに使用します。
P4のプログラムは記録用で、プログラムスタートと同時にデータを記録しますので、スタート前に〈FA-3〉とカセットテープレコーダに接続し、マイク用端子とリモート用端子を差し込んで、「録音」状態にしておいてください。
P5のプログラムは再生用ですので、スタート前に、〈FA-3〉とカセットテープレコーダーに接続し、イヤホン用端子とリモート端子を差し込み、「再生」状態にしておいてください。
なお、記録するときは新しいテープを、再生するときはデータの記録してあるテープをセットしておいてください。

〈ポイント〉

このプログラムでは全部で1,091ステップをプログラムとして使用していますので、データ数(たて×よこ)はRC-2使用時は59個以内、RC-4使用時は315個以内となっています。もし、もっと多くのデータを扱いたい場合は、不要なプログラムを記憶させずに残りステップ数に応じてP0の行番号80の"59"を変更してください。
P3に入れている計算用のプログラムは、たて計とよこ計、そして総合計を求めるものですので、他の計算を行ないたい方は、このプログラムを変更してみるのもよいでしょう。

P0

```
 10 PRINT "New [Y/N
    ]?";
 20 K$= KEY$: IF K$
    ="Y" THEN PRINT
    : GOTO 50
 30 IF K$="" THEN 2
    0
 40 PRINT : GOTO 21
    0
 50 CLEAR
 60 INPUT "TATE",Y
 70 INPUT "YOKO",X
 80 IF Y*X>59 THEN
    60
 90 DEFM X*Y
100 FOR I=1 TO Y
110 FOR J=1 TO X
120 PRINT "(";I;","
    ;J;")";: INPUT
    $
130 IF $>"≠" THEN I
    F $<"π" THEN 18
    0
140 IF $≠"=" THEN 1
    10
150 IF J-1>0 THEN J
    0J-1: GOTO 120
160 IF I-1<1 THEN 1
    20
170 I=I-1:J=X: GOTO
     120
180 Z((I-1)*X+J)= V
    AL($)
190 NEXT J
200 NEXT I
210 PRINT "END"
```

270ステップ

P1

```
 10 PRINT "Printer[
    Y/N]";
 20 K$= KEY$: IF K$
    ="" THEN 20
 30 PRINT
 40 IF K$="Y" THEN
    MODE 7: PRINT "
    DATA"
 50 FOR I=1 TO Y
 60 FOR J=1 TO X
 70 PRINT "(";I;","
    ;J;")";Z((I-1)*
    X+J)
 80 NEXT J
 90 IF K$="Y" THEN
    PRINT " "
100 NEXT I
110 MODE 8
120 PRINT "END"
```

151ステップ

P2

```
 10 INPUT "TATE",$
 20 IF $="=" THEN P
    RINT "END";: EN
    D
 30 IF $>"≠" THEN I
    F $<"π" THEN 50
 40 GOTO 10
 50 INPUT "YOKO",P
 60 O= VAL($)
 70 PRINT "(";O;","
    ;P;")";Z((O-1)*
    X+P);: INPUT $
 80 IF $="=" THEN 1
    0
 90 IF $="+" THEN 1
    40
100 IF $="-" THEN 1
    60
110 IF $>"≠" THEN I
    F $<"π" THEN 13
    0
120 GOTO 70
130 Z((O-1)*X+P)= V
    AL($)
140 IF P+1>X THEN O
    =O+1:P=0: IF O>
    Y THEN O=1:P=1:
     GOTO 70
150 P=P+1: GOTO 70
160 IF P-1<1 THEN O
    =O-1:P=X+1: IF
    O<1 THEN O=Y:P=
    X: GOTO 70
170 P=P-1: GOTO 70
```

293ステップ

P3

```
10 PRINT "Printer[
  Y/N]";
20 K$= KEY$: IF K$
  ="" THEN 20
30 IF K$="Y" THEN
  MODE 7
40 PRINT
50 PRINT "YOKO TOT
  AL"
60 FOR I=1 TO Y
70 A=0
80 FOR J=1 TO X
90 A=A+Z((I-1)*X+J
  )
100 NEXT J
110 PRINT "(";I;")"
  ;A
120 NEXT I
130 PRINT "TATE TOT
  AL"
140 B=0
150 FOR J=1 TO X
160 A=0
170 FOR I=1 TO Y
180 A=A+Z((I-1)*X+J
  )
190 NEXT I
200 PRINT "(";J;")"
  ;A
210 B=B+A
220 NEXT J
230 PRINT "GRAND TO
  TAL"
240 PRINT B
250 MODE 8
```

263ステップ

P4

```
10 PRINT "DATA PUT
  ";
20 PUT "DATA"X,Y
30 PUT Z(1),Z(X*Y)
40 PRINT "→END"
```

53ステップ

P5

```
10 PRINT "DATA GET
  ";
20 GET "DATA"X,Y
30 DEFM X*Y
40 GET Z(1),Z(X*Y)
50 PRINT "→END"
```

60ステップ

計 1090ステップ

カーレース

このゲームは変化するコースを右に左にとハンドルを切り、フェンスにぶつからないように長い距離を走りぬくレースです。

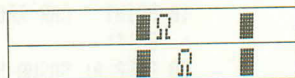
■プログラムリスト

```
 10 PRINT " CAR RAC
    E !";
 20 BEEP 0: GOSUB 5
    00
 30 PRINT "HI-SCO:"
    ;S;"km";
 40 GOSUB 500
 50 X=6:Y=3:Z=9:T=0
    :C=0
 60 PRINT
 70 PRINT CSRY;"█";
     CSRX;"Ω"; CSRZ
    ;"█";
 80 IF X= INTY THEN
     GOSUB 600
 90 IF X= INTZ THEN
     GOSUB 600
100 T=T+1
110 $= KEY$
120 IF $="4" THEN X
    =X-1
130 IF $="6" THEN X
    =X+1
140 BEEP 0
150 R= RAN#*.9
160 IF RAN#>.5 THEN
     R=-R
170 IF Z+R≥12 THEN
    R=0
180 Q= RAN#*.8
190 IF RAN#>.5 THEN
     Q=-Q
200 IF Y+Q<0 THEN Q
    =0
210 IF Z-Y<3 THEN 2
    30
220 Z=Z+R:Y=Y+Q
230 GOTO 60
500 REM TIME
510 FOR U=1 TO 100:
     NEXT U
520 PRINT
530 BEEP 0
540 RETURN
600 REM CRASH
610 FOR I=1 TO 10
620 PRINT CSRX;"*";
630 BEEP 1
640 PRINT CSRX;"Ω";
650 NEXT I
660 PRINT CSR0;"<<C
    RASH !!>>";
670 GOSUB 500
680 PRINT "SCORE:";
    T*3;"km";
690 GOSUB 500
700 X=6:Y=3:Z=9
710 C=C+1
720 IF C<3 THEN RET
    URN
730 T=T*3
740 IF T>S THEN S=T
750 PRINT
760 $="GAME OVER !!
    "
770 FOR I=1 TO 12
780 PRINT MID$(I,1)
    ;: BEEP 1
790 NEXT I
800 END
```

計 540steps

■**ゲーム説明**

使うキーは④と⑥だけで、車を左に動かすときは④キーを押し、車を右に動かすときは⑥キーを押します。

左右のフェンスが動き、コースが広くなったり狭くなったりします。このフェンスにぶつからないようにうまくキーを操作してください。

車は3台あります。フェンスにぶつかるとクラッシュし、その時点の走行距離が表示されます。

<<CRASH !!>>
SCORE: 45km

クラッシュは2回までだいじょうぶですが、3回するとゲームオーバーとなります。

<<CRASH !!>>
SCORE: 234km
GAME OVER !!

潜水艦を撃沈せよ

このゲームは海上を航行する駆逐艦をうまく操縦し、敵の潜水艦を撃沈します。ただし、駆逐艦に備えてあるソナーは性能が悪く、真下に潜水艦がきたときしか、反応しませんし、深度もわかりません。また、駆逐艦の燃料は少ししかつめません。

このような状況下で、敵の魚雷をかわしながら戦います。

■プログラムリスト

```
10 PRINT " <SUBMARINE>";
20 BEEP : GOSUB 500
30 PRINT "HI-SCO:";T;
40 BEEP : GOSUB 500
50 X=4:S=100:R=0:N=0:L=3
60 FOR I=0 TO 2
70 A(I)= INT( RAN#*10):D(I)= INT( RAN#*10)
80 NEXT I
90 FOR K=0 TO 2
100 PRINT
110 S=S-1
120 IF S<20 THEN BEEP 1
130 IF S<0 THEN 370
140 PRINT "■■■■■■■■■■"; CSRX;"♠";
150 IF A(K)=X THEN PRINT CSR11;"*";
160 $= KEY$: IF $="" THEN 200
170 IF $="Z" THEN X=X-1: IF X<0 THEN X=0: GOTO 200
180 IF $="X" THEN X=X+1: IF X>9 THEN X=9: GOTO 200
190 IF $≧"0" THEN IF $≦"9" THEN GOSUB 600
200 IF A(K)<0 THEN 360
210 IF RAN#<.8 THEN 300
220 A(K)=A(K)-1
230 IF RAN#>.5 THEN A(K)=A(K)+2
240 D(K)=D(K)-1
250 IF RAN#>.5 THEN D(K)=D(K)+2
260 IF A(K)<0 THEN A(K)=0
270 IF A(K)>9 THEN A(K)=9
280 IF D(K)<0 THEN D(K)=0
290 IF D(K)>9 THEN D(K)=9
300 IF X=A(K) THEN IF N=0 THEN IF RAN#>.8 THEN N=1:M=D(K)
310 IF N=0 THEN 350
320 PRINT CSR11;"↑";: BEEP
330 IF M<0 THEN N=0: IF X=A(K) THEN GOSUB 900
340 M=M-1
350 GOTO 100
360 NEXT K
370 PRINT
380 IF S>0 THEN R=R+S
390 PRINT "SCORE:";R;
400 IF T<R THEN T=R: FOR I=1 TO 10: BEEP 1: NEXT I
410 IF S<0 THEN 440
420 IF L<1 THEN 440
430 END
440 GOSUB 500
450 $="GAME OVER !!"
460 FOR I=1 TO 12
470 PRINT MID$(I,1);: BEEP 1
480 NEXT I
490 END
500 REM SUBTIME
510 FOR U=1 TO 100: NEXT U
520 PRINT
530 RETURN
600 REM FIRE
```

```
610 BEEP
620 IF A(K)=X THEN
    IF D(K)= VAL($)
     THEN 650
630 IF A(K)=X THEN
    IF ABS(D(K)- VA
    L($))<2 THEN 71
    0
640 RETURN
650 FOR I=1 TO 10
660 PRINT CSR11;"*"
    ;: BEEP 1
670 PRINT CSR11;"+"
    ;: BEEP 0
680 NEXT I
690 A(K)=-1:R=R+ IN
    T( RAN#*5+1)*10
    0:S=S+50
700 RETURN
710 FOR I=1 TO 5
720 PRINT CSR11;"Ω"
    ;: BEEP 0
730 NEXT I
740 RETURN
900 REM DEAD
910 FOR I=1 TO 10
920 PRINT CSRX;"x";
    : BEEP 1: PRINT
     CSRX;"♠";
930 NEXT I
940 L=L-1
950 IF L<1 THEN PRI
    NT : GOTO 380
960 RETURN

    計 999steps
```

■ゲーム説明

海の中は次のようになっています。

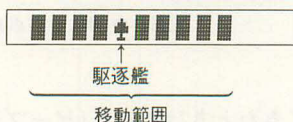
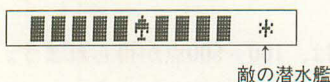
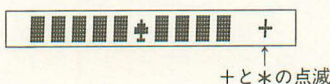
駆逐艦を動かすには、左に移動するときは[Z]キーを、右に移動するときは[X]キーを押します。潜水艦を攻撃する爆雷は深度を指定しなければなりません。深度は[0]～[9]のキーを押して決めます。

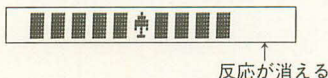
駆逐艦も敵の潜水艦も3隻づつあります。潜水艦を3隻とも沈めればゲーム終了となり、得点が表示されます．先に駆逐艦が3隻とも沈められたり、燃料が切れたときはゲームオーバーとなります。

スタート後はタイトルとハイスコアーを表示し、ゲームにはいります。

RUN EXE

<SUBMARINE>
HI-SCO: 252

まず、駆逐艦と移動範囲が表示されます。

Zキーまたは Xキーを使って、駆逐艦を左右に動かしますと、ソナーに敵の反応が出てきます。

深度はわかりませんが、真下にいます。
ここだと思う深度のキー(0~9)を押して、爆雷を発射します。
音とともに爆雷が沈んでいきます。潜水艦に命中したときは、潜水艦が爆発します。

命中しないときでも近い場合(深度が±1)は、ソナーの反応がかわります。

敵の潜水艦は深度をかえながら左右に逃げますので、見失わないようにおいかけます。潜水艦が真下から逃げたときは、ソナーの反応が消えます。

また、敵の潜水艦は逃げてばかりではなく、時々魚雷で攻撃してきます。

魚雷の攻撃を受けたときは、一目散に逃げます。敵の魚雷は高性能で、少しぐらい逃げても追ってくることがあります。

なお、燃料が少なくなってきたときは断続音(ビープ音)で知らせてくれます。燃料の補給はできず、なくなったときはゲームオーバーとなります。ただし、敵の潜水艦を撃沈したときは、少しだけ敵の燃料をとることができます。このようにして、なるべく早く、敵の潜水艦を沈めてください。

〔得点の方法〕

潜水艦を撃沈したときは、100～500点が得られます。
その他に、残り燃料分の得点が加算されます。

陸上競技

このゲームは3つの種目からできています。

1 (P0)：100m競走
2 (P1)：走り幅飛び
3 (P2)：ハードル

これ等のプログラムは独立したプログラムエリアにしてあります。
順番としてまずP0の100m競争から始め、一定レベル以上の成績を上げれば、次の競技に進めます。最後のハードルを完走した場合には総合点も表示されます。

■プログラムリスト

```
P0
  10 X=0:Y=0:W=0:Z=0
     :K=0:B=100
  20 PRINT "HI-SCO";
     Q;"s";
  30 V=B: GOSUB #9:
     BEEP 1
  40 IF KEY$≠"" THEN
      GOSUB #7: GOTO
      30
  50 PRINT CSRX;"Ω";
      CSR11;"!";
  60 FOR I=1 TO 5
  70 IF KEY$≠"" THEN
      W=W+.2
  80 Z=Z+1
  90 NEXT I
 100 PRINT CSRX;" ";
 110 FOR I=1 TO 5
 120 IF KEY$≠"" THEN
      W=W-.2
 130 NEXT I
 140 X=X+W:W=0
 150 IF INTX≠Y THEN
     BEEP :Y= INTX
 160 IF X<0 THEN X=0
 170 IF X<11 THEN 50
 180 PRINT : BEEP 1
 190 D= RND(Z/12,-3)
 200 PRINT "TIME:";D
     ;"s";
 210 IF Q=0 THEN Q=D
 220 IF D<Q THEN Q=D
     : GOSUB #6
 230 GOSUB #9
 240 IF D≥15 THEN #8
 250 PRINT "NEXT GAM
     E ?";
 260 IF KEY$="" THEN
      260
 270 GOTO #1
```

343steps

```
P1
  10 MODE 4:K=0
  20 PRINT
  30 PRINT "HI-SCO";
     P;"m";
  40 V=B: GOSUB #9
  50 W=0
  60 BEEP 1
  70 FOR X=0 TO 11 S
     TEP .5
  80 PRINT CSRX;"Ω";
      CSR11;"_";
  90 $= KEY$
 100 IF $≤"Z" THEN I
     F $≥"A" THEN W=
     W+1
 110 IF $≥"0" THEN I
     F $≤"9" THEN GO
     TO 200
 120 V=20-W
 130 GOSUB #9: BEEP
 140 NEXT X
 150 FOR M=1 TO 5
 160 $= KEY$
 170 IF $≥"0" THEN I
     F $≤"9" THEN 20
     0
 180 NEXT M
 190 GOSUB #7: GOTO
     40
 200 BEEP 1
 210 FOR J=1 TO 80
 220 IF KEY$="" THEN
      240
 230 NEXT J
 240 BEEP
 250 R=W/2*(6-M)* CO
     S ABS(45-J)/6
 260 FOR X=0 TO R ST
     EP .5
 270 PRINT CSRX;"o";
 280 V=10: GOSUB #9
 290 BEEP 1
 300 NEXT X
 310 BEEP
 320 PRINT CSRR;"Ω";
 330 V=B: GOSUB #9
 340 E= RND(R,-3)
 350 IF E<2 THEN GOS
     UB #7: GOTO 40
```

```
360 PRINT "SCORE:";
    E;"m";
370 IF P<E THEN P=E
    : GOSUB #6
380 V=B: GOSUB #9
390 IF E<7 THEN #8
400 PRINT "NEXT GAM
    E ?";
410 IF KEY$="" THEN
     410
420 GOTO #2
```

456steps

P2

```
 10 $="    1   1
    1   1   1":X=0:
    K=0:Y=2:W=0:Z=0
 20 PRINT : PRINT "
    HI-SCO";O;"s";
 30 V=B: GOSUB #9
 40 BEEP 1
 50 IF KEY$≠"" THEN
     GOSUB #7: GOTO
    30
 60 PRINT CSR0; MID
    $(Y,11);
 70 PRINT CSRX;"Ω";
 80 Z=Z+1
 90 A$= KEY$
100 IF A$≦"9" THEN
    IF A$≧"0" THEN
    H=1: BEEP 1: PR
    INT CSR0;"μ";
110 IF Y=1 THEN Y=Y
    +1: IF H≠1 THEN
    Z=Z+3: GOSUB #
    7
120 IF A$≦"Z" THEN
    IF A$≧"A" THEN
    Y=Y+1:W=W+1: BE
    EP
130 IF Y>4 THEN Y=1
140 H=0
150 IF W<30 THEN 60
160 PRINT CSR0;"1
     1   1   !";
170 PRINT CSRX;"Ω";
180 Z=Z+1
190 A$= KEY$
200 IF A$≦"9" THEN
    IF A$≧"0" THEN
    H=1: BEEP 1: PR
    INT CSRX;"μ";
210 IF FRAC(X/4)=0
    THEN X=X+1: IF
    H≠1 THEN Z=Z+3:
     GOSUB #7
220 IF A$≦"Z" THEN
    IF A$≧"A" THEN
    X=X+1: BEEP
230 H=0
240 IF X<12 THEN 16
    0
250 BEEP : PRINT
260 F= RND(Z/1.1,-2
    )
270 PRINT "TIME:";F
    ;"s";
280 IF O=0 THEN O=F
290 IF F<O THEN O=F
    : GOSUB #6
300 GOSUB #9
310 IF F>60 THEN #8
320 R= INT( COS(D*3
    )*200+ SIN(E*3)
    *200+ COS(F*3)*
    200)
330 PRINT "TOTAL SC
    ORE";R;" points
     ";
340 IF T=0 THEN T=R
350 IF T<R THEN T=R
    : GOSUB #6
```

603steps

P6

```
 10 FOR I=1 TO 10:
    BEEP 1: BEEP :
    NEXT I
 20 RETURN
```

22steps

P7

```
 10 PRINT : PRINT "
    FOUL !!";: BEEP
    : BEEP
 20 IF K=0 THEN K=1
    : RETURN
 30 GOTO #8
```

39steps

P8

```
 10 PRINT
 20 $="GAME OVER !!
    "
 30 FOR I=1 TO 12
 40 PRINT MID$(I,1)
    ;: BEEP
 50 NEXT I
```

20steps

P9

```
 10 FOR U=1 TO V: N
    EXT U
 20 PRINT
 30 RETURN
```

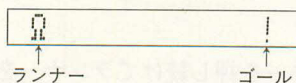
50steps

計 1533steps

P0：100m競争
P1：走り幅飛び
P2：ハードル
P6：BEEP効果音
P7：ファウル処理
P8：ゲームオーバー処理
P9：一定時間停止用

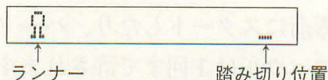
■ゲーム説明

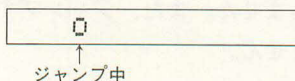
RUN[EXE]または[SHIFT][P0]でスタートさせます。

100m競争

ランナーが点滅しますので、表示しているときにいずれかのキーを押すと前進(右に進む)します。ランナーが消えているときに押すと後退(左に進む)します。

"TIME"(経過時間)が15秒以内ですと、次の走り幅飛びに進めます。

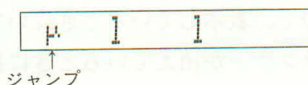
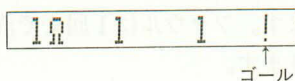
なお、ランナーが表示される前にキーを押しますと「FOUL!!」(ファウル)となり、やり直しとなります。ファウルは1回まで許されますが、2回行なうとゲームオーバーとなります。

走り幅飛び

アルファベットキー([A]~[Z])を押すとランナーの速度が増します。キーを押さなくても進みますが、加速がつかないとジャンプに失敗します。

ランナーが踏み切り位置にきたときに、数値キー([0]~[9])を押します。このとき、数字キーを押している時間により飛び出す角度がかわりますので、押す時間は短かすぎても長すぎてもいけません。

ジャンプした後はランナーが飛びます。踏み切り位置をすぎでもジャンプしないときやジャンプに失敗したときはファウル(FOUL!!表示)となり、1回まで許されます。

なお、ジャンプした距離が7m以下の場合は失格となり、次のハードルには進めません。

ハードル

まず，アルファベットキーを押し続けてランナーを走らせます。ランナーがハードルに近づいたらタイミング良く数字キーを押してジャンプさせます。

ハードルをいくつか飛びこえていくと、ゴールが見えてきます。

ランナーが表示される前にスタートしたり、ハードルをたおしたりするとファウルとなります。ファウルは1回まで許されます。
経過時間が60秒以上ですと、失格となり総合得点は表示されません。

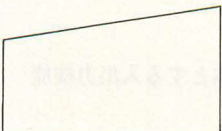
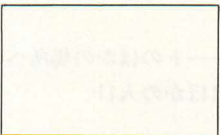
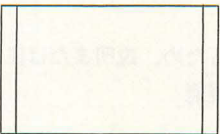
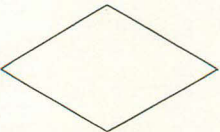
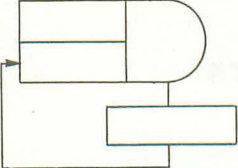
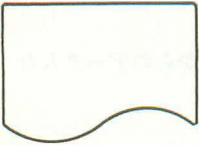
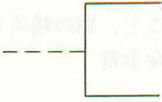
●使うキー

100m競争で使うキーは[MODE]、[⇦]、[⇨]、[SHIFT]、[FUNC]、[AC]、[DEL]、[STOP]、[EXE]、[MEMO]以外なら、どのキーを押してもかまいません。
走り幅飛びとハードルでは、ランナーが走るときは[A]～[Z]のアルファベットキーならどれでもかまいません。また、ジャンプするときは[0]～[9]の数字キーならどれでもかまいません。

エラーメッセージ一覧表

エラーコード	意　味	エラー原因	対　策
1	メモリーオーバーまたはシステムスタックオーバー	●ステップ数不足でプログラムが書き込めない。またはメモリーが増設できない。 ●計算式が複雑すぎてスタックオーバーしている。	●不要なプログラムをクリアするか、メモリー数を減らす。 ●数式を分割して簡単にする。
2	構文エラー	●プログラム等に書式上の誤りがある。 ●代入文等で左辺の型式と右辺の型式が異なる。	●入力したプログラム等の誤りを修正する。
3	数学的エラー	●数式の演算結果が10^{100}以上の場合。 ●数値関数の入力範囲外の場合。 ●結果が不定または不能となる場合。	●演算式またはデータを修正する。 ●データを判断する。
4	未定義エラー	●GOTO文、GOSUB文の指定行番号がない。 ●READ文またはREAD#文に対するデータが不足している。	●指定行番号を修正する。 ●データ数を正しくする。
5	引数エラー	●引数を必要とするコマンド、関数において、引数が入力範囲外の場合。	●引数の誤りを修正する。
6	変数エラー	●増設されていないメモリーを使おうとした。 ●同一メモリーを数値変数と文字変数に同時に使おうとした。	●適切にメモリーを増設する。 ●同時に同一メモリーを文字変数、数値変数として使わないようにする。
7	ネスティングエラー	●サブルーチン実行中以外でRETURN文が出てきた場合。 ●FORループ中以外でNEXT文が出てきたり、FOR文に対するNEXT文の変数が異なる場合。 ●サブルーチンのネスティングが8段をこえた場合。 ●FORループのネスティングが4段をこえた場合。	●不必要なRETURN文やNEXT文を取る。 ●サブルーチンやFOR･NEXTループをレベル内にする。

エラーコード	意　味	エ ラ ー 原 因	対　策
8	パスワードエラー	●パスワードが設定されているときに、異なるパスワードを入力した。 ●パスワードが設定されているのに、LISTやNEWなどの禁止コマンドを実行した。	●正しいパスワードを入力して、パスワードを解除する。
9	オプションエラー	●テープレコーダが接続されていないのに、SAVEまたはPUTコマンドを実行した場合。 ●LOADまたはGETコマンドにより入力された信号がわれており、読み込めない場合。 ●プリンタの充電が十分でない場合。 ●プリンタの紙づまり。	●テープレコーダを接続する。 ●テープレコーダの再生ボリュームを下げる。 ●テープレコーダのTONEを中位に調整する。 ●テープをかえる。 ●テープレコーダのヘッドをクリーニングする。 ●プリンタを充電する。 ●プリンタの紙づまりを直す。

キャラクター表

	スペース	+	−	*	/	↑	!	"	#	$	>	≧	=	≦	<	≠
数字	0	1	2	3	4	5	6	7	8	9	.	π	)	(	Ē	E
英大文字	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Q	R	S	T	U	V	W	X	Y	Z						
英小文字	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
	q	s	r	t	u	v	w	x	y	z						
記号	?	,	;	:												
グラフィック記号	○	Σ	°	△	@	×	÷	♠	←	♥	♦	♣	μ	Ω	↓	→
	%	¥	□	[	&	_	'	·	]	■	＼					

表は左から右へ、次の段の左から右へと小さい順に並んでいます。

一番小さいキャラクターは、スペースで、一番大きいキャラクターは、グラフィク記号の＼（SHIFT P と押すと表示される）です。

フローチャートの主な記号

記号	名称	意味
	端子	開始、終了等
	手操作入力	キーボードからのデータ入力
	出力	出力の機能
	処理	あらゆる種類の処理機能
	定義済み処理	サブルーチンなど、別の場所で定義されている命令群
	判断	いくつかの択一的径路のうち、どの径路をとらせるかを決める判断

記　　号	名　　称	意　　味
	FOR・NEXTループ	FOR文とNEXT文の間で、一定回数分処理を行なう。
	書　　類	書類を媒体とする入出力機能
	流　れ　線	記号を結びつける機能
	結　合　子	フローチャートのほかの場所への出口またはほかの入口
	注　　釈	明瞭にするため、説明または注意を加える機能

配列変数表

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
B	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
C	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
E	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
F	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
G	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
J	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12
O	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11
P	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10
Q	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9
R	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8
S	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7
T	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6
U	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5
V	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4
W	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3
X	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2
Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1
Z	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

表の見方

縦に並んでいるＡ～Ｚを配列変数名として使用した場合、通常のＡ～Ｚの変数とどこから重なるかを見ます。

例）H(0)～H(9)→H～Q

規　格

型　　　式：PB-410/FX-720P
基本計算機能：負数、指数、カッコを含む四則計算(加減・乗除の優先順位判別機能つき)
組込関数機能：三角・逆三角関数(角度単位は度・ラジアン・グラジアン)、対数・指数関数、開平、べき乗、整数化、整数部除去、絶対値、符号化、有効桁数指定、小数点以下指定、10↔60進変換、乱数、π
コマンド：INPUT、PRINT、GOTO、ON～GOTO、FOR・NEXT、IF～THEN、GOSUB、ON～GOSUB、RETURN、READ、DATA、RESTORE、STOP、END、RUN、LIST、LIST ALL、MODE、SET、CLEAR、NEW、NEW ALL、DEFM、PASS、REM、BEEP、LET、SAVE、SAVE ALL、LOAD、LOAD ALL、PUT、GET、VERIFY
NEW#、LIST#、LOAD#、SAVE#、READ#、WRITE#、RESTORE#
プログラム関数：KEY$、CSR、LEN、MID$、VAL、STR$
計算範囲：$\pm1\times10^{-99}\sim\pm9.999999999\times10^{99}$および0、内部演算は仮数部12桁を使用
プログラム言語：BASIC(ベーシック)
RAM容量：RC-2使用時　2Kバイト　RC-4使用時　4Kバイト　(システムエリア272バイト、固定変数エリア208バイト含む)
組込プログラム数：最大10組(P0～P9)
メモリー数：標準26メモリー、および専用文字変数($)
スタック数：サブルーチン　8段・FOR・NEXTループ　4段
数値　6段・演算子　12段
表示桁数および内容：仮数部10桁(負符号含む)、または仮数部8桁(負数7桁)+指数部2桁、内容　EXT、S、RUN、WRT、DEG、RAD、GRA、TR、PRT、STOPの各状態表示付
表示素子：12桁ドットマトリクス液晶
主要素子：C-MOS VLSI他
電　　源：リチウム電池(CR-2032)2個使用
消費電力：最大0.03W
電池寿命(連続使用)：本体のみ 約140時間
オプション使用時 約70時間
RAMカード(計算機外保存時) RC-2 約2年間
RC-4 約1年間
オートパワーオフ：約6分
使用温度：0℃～40℃
大きさ・重さ：本体 幅165 奥行82 高さ14.3㎜、186g(電池・RAMカード込み)
RAMカード 幅60 奥行50 厚さ3.8㎜、26g(電池込み)
付属品：ソフトケース

コマンド索引

カシオ計算機株式会社営業本部

東京都新宿区西新宿2—6　新宿住友ビル
(〒160)　　☎03-347-4811(代表)

カシオ計算機サービスセンター

拠点	電話	郵便番号	住所
旭川	0166-23-8580	〒070	旭川市七条通り8丁目
札幌	011-231-2343	〒060	札幌市中央区南一条西12丁目
釧路	0154-24-8575	〒085	釧路市光陽町6—1
青森	0177-22-7466	〒030	青森市勝田2—1—12
秋田	0188-33-6211	〒010	秋田市中通り6—1—15
盛岡	0196-24-2502	〒020	盛岡市本町通り3—19—6
仙台	0222-27-1404	〒980	仙台市一番町2—3—32
山形	0236-42-8018	〒990	山形市あこや町3—14—39
郡山	0249-33-5172	〒963	福島県郡山市香久池2—16—6
宇都宮	0286-34-0395	〒320	宇都宮市西大寛2—1—3
前橋	0272-53-3000	〒371	前橋市元総社町92—5
水戸	0292-25-6985	〒310	水戸市中央1—2—20
埼玉	0486-66-8567	〒330	大宮市大成町4—83
千葉	0472-43-1751	〒260	千葉市登戸町2—276
東京	03—862-4141	〒101	千代田区神田佐久間町2—23
中央	03—583-4111	〒106	港区六本木2—3—6
城南	03—787-3721	〒145	大田区上池台1—1—6
城西	03—376-3221	〒160	新宿区西新宿4—2—18
多摩	0425-23-3531	〒190	立川市錦町3—2—25
横浜	045-211-0821	〒231	横浜市中区弁天通り6—85
新潟	0252-41-4105	〒950	新潟市米山3—1—5
長野	0262-28-9360	〒380	長野市岡田町30—20
甲府	0552-37-6371	〒400	甲府市城東2—22—11
静岡	0542-81-8085	〒420	静岡市西中原1—4—35
浜松	0534-64-1658	〒435	浜松市西塚町324
豊橋	0532-53-2515	〒440	豊橋市魚町5
名古屋	052-263-0454	〒460	名古屋市中区栄4—6—15
岐阜	0582-62-0145	〒500	岐阜市鷹見町8
三重	0592-27-5066	〒514	津市鳥居町191
富山	0764-22-2251	〒930	富山市白銀町2—1
金沢	0762-37-8511	〒920	金沢市諸江町下丁93—1
京都	075-351-1161	〒600	京都市下京区五条通り堀川東入ル
大阪	06—362-8181	〒530	大阪市北区南森町2—1—20
和歌山	0734-31-7807	〒640	和歌山市九家の丁5
神戸	078-392-4123	〒650	神戸市中央区北長狭通り4—4—18
岡山	0862-41-8471	〒700	岡山市西古松西町8—21
福山	0849-24-2830	〒720	福山市南本庄町2—130
広島	082-263-1090	〒730	広島市南区稲荷町4—1
山口	0835-22-6164	〒747	防府市戎町1—10—16
高松	0878-62-5240	〒760	高松市亀岡町9—16
松山	0899-45-2234	〒790	松山市平和通り1—1—5
福岡	092-411-2684	〒812	福岡市博多区博多駅南1—2—15
長崎	0958-61-8084	〒852	長崎市宝栄町2—26
熊本	096-367-0650	〒862	熊本市健軍4—1—5
鹿児島	0992-56-3575	〒890	鹿児島市上荒田町30—18